IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s)	:	Jones, John David
Application #	:	10/734,010
Confirmation #	:	2351
Filed	:	11 December 2003
Application Title	:	Systems, Devices, and Methods for Acceptance Testing a Fieldbus Component Configuration Program
Art Unit #	:	2857
Latest Examiner	:	Assouad, Patrick J.

Assistant Commissioner for Patents
Washington, DC 20231

DECLARATION UNDER 37 C.F.R. § 1.131

I, John David Jones, a citizen of the United States, whose full post office address is 127 Burton France Rd. Johnson City, TN 37604 sincerely declare that:

1. I am an inventor of the above-identified U.S. Patent Application Serial Number 10/734,010 (the Application).
2. Upon information and belief, as will be detailed below, the attached documents evidence that the subject matter of each of the claims of the Application was conceived in the United States and pursued with diligence from a date prior to the effective filing date (11 December 2002) of Published United States Patent Application No. 10/316,720 (Cassiolato), which has been cited in a rejection against certain claims of the Application.
3. Upon information and belief, Document A, which is titled "Invention Disclosure", to which I provided my dated signature on 19 November 2002, and which was stamped

as received by the Intellectual Property Department of my employer, Siemens Corporation, on 3 December 2002, briefly describes the "purpose(s), and likely field(s) of use of this invention", and evidences conception and that the Application was pursued with diligence from at least 19 November 2002.

4. Upon information and belief, Document B, which is titled "AS-I Wizard", was included with Document A when I signed Document A on 19 November 2002, contains a description of the subject matter claimed in the Application, and evidences conception and that the Application was pursued with diligence from at least 19 November 2002.
5. Upon information and belief, Document C, which is titled "AS-I Wizard Test Plan", was included with Document A when I signed Document A on 19 November 2002, contains a description of the subject matter claimed in the Application, and evidences conception and that the Application was pursued with diligence from at least 19 November 2002.
6. Upon information and belief, the Application claims priority to United States Provisional Patent Application No. 60/436,261, which was filed on 23 December 2002, and which included a copy of Documents B and C, thus containing a description of the subject matter claimed in the Application, and evidencing conception and that the Application was pursued with diligence to 23 December 2002, the effective file date of the Application.

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code and that willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Signed this 10th day of May, 2005

John David Jones

Document A — **INVENTION DISCLOSURE** — Page 1 of ____

Return this completed form, with any necessary attachments, to:
SIEMENS CORPORATION
Intellectual Property Department (IPD)
186 Wood Avenue South
Iselin, NJ 08830
phone +1-732-321-3026
fax +1-732-321-3014
PLEASE KEEP A COPY FOR YOUR RECORDS

THIS SECTION FOR IPD USE ONLY
Receipt RECEIVED
DEC 03 2002
INTELLECTUAL PROPERTY DEPARTMENT
Division/Dept Code ____
Cost Center Code 2106
Docket No. 02E 19812US
IP Attorney BMR ____

1. Your Div.'s Project # ____; Invention Title SYSTEM AND METHOD FOR TESTING AND SIMULATING DISTRIBUTED I/O SYSTEMS

2. INVENTOR

At the time the invention was conceived, this inventor was
☐ *an employee of a German legal entity (e.g. Siemens AG)*
☒ *an employee of a non-German entity (e.g. Siemens Automotive Corporation or Siemens Canada Limited)*
If you are not sure, contact IPD at +1-732-321-3026

Full Name JOHN DAVID JONES
Home Address 127 BURTON FRANCE RD.
JOHNSON CITY, TN 37604

Citizen of UNITED STATES OF AMERICA
Soc. Sec. # 413-29-2839
COMPANY/LOCATION INFORMATION
Employer SIEMENS ENERGY & AUTOMATION
Division MICROSYSTEMS
Address ONE INTERNET PLAZA
JOHNSON CITY TN 37604

Work telephone (423) 262-2626
Fax (423) 262-2174
Email JONES.DAVE@SIEMENS.COM

Signature John David Jones
Date 19-Nov-2002

3. CO-INVENTOR (if any)
(use additional forms for more co-inventors)

At the time the invention was conceived, this inventor was
☐ *an employee of a German legal entity (e.g. Siemens AG)*
☒ *an employee of a non-German entity (e.g. Siemens Automotive Corporation or Siemens Canada Limited)*
If you are not sure, contact IPD at +1-732-321-3026

Full Name STEVE HAUSMAN
Home Address 2322 CAMELOT CIRCLE
JOHNSON CITY TN 37604

Citizen of UNITED STATES OF AMERICA
Soc. Sec. # 359-56-6796
COMPANY/LOCATION INFORMATION
Employer SIEMENS ENERGY & AUTOMATION
Division MICROSYSTEMS
Address ONE INTERNET PLAZA
JOHNSON CITY TN 37604

Work telephone (423) 262-2154
Fax (423) 262-2174
Email STEVE.HAUSMAN@SIEMENS.COM

Signature Steven Hausman
Date 19-Nov-2002

4. Purpose(s), and likely field(s) of use of this invention TO TEST, SIMULATE, DEBUG AND DESIGN DISTRIBUTED I/O SYSTEMS SUCH AS ASI (ACTUATOR SENSOR INTERFACE). IT MINIMIZES START UP COSTS, INCLUDING TIME REQUIREMENTS IN DEVELOPING, DEBUGGING AND TESTING A DISTRIBUTIVE NETWORK. ONILIMITS FIELDS OF USE INCLUDING INDUSTRIAL AUTOMATION AND PROCESS CONTROL FOR INDUSTRY USERS IN THE PHARMACEUTICAL, PETROLEUM, CHEMICAL, ECT. (ALL DISTRIBUTED NETWORKS).

6. Where is this invention described? List all documents containing aspects of this invention (e.g. engineer notebook date/page, internal status report, test logs, physical prototypes, customer presentation, etc) SEE ATTACHED

IF THE ANSWERS TO ANY OF THESE QUESTIONS CHANGE BEFORE THIS INVENTION IS FILED AS A PATENT APPLICATION WITH THE GOVERNMENT, MAKE SURE TO SEND AN UPDATE TO SIEMENS IPD.

11. WITNESSED AND UNDERSTOOD BY:

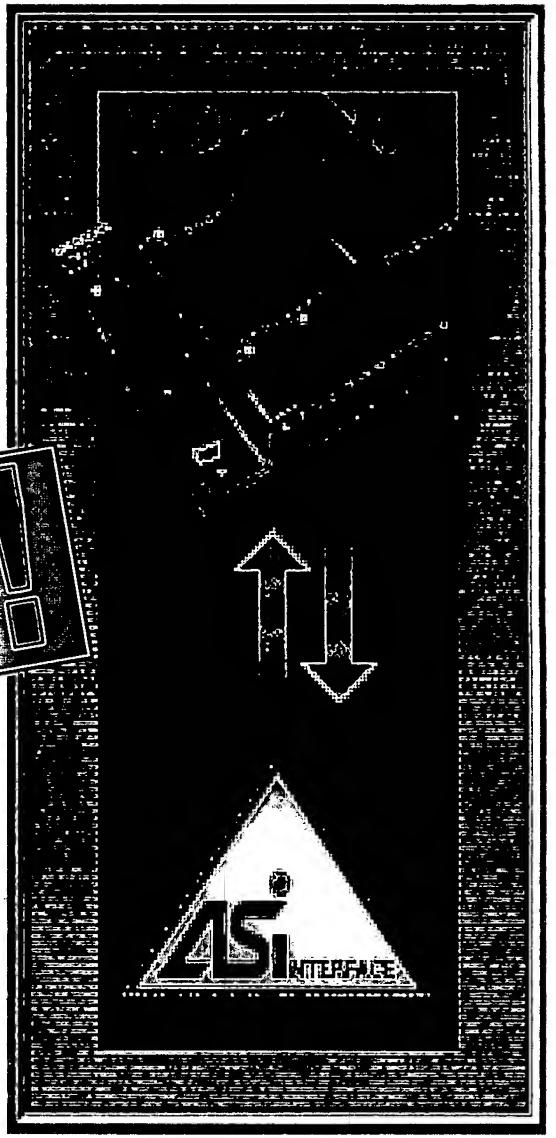
Signature ____ Company Name ____
Name ____ Company Address ____
Date ____

THIS INVENTION DISCLOSURE IS SIEMENS PROPRIETARY INFORMATION, AND IS CONFIDENTIAL. THIS DISCLOSURE AND ITS SUBJECT MATTER MAY NOT BE DISCLOSED IN WHOLE OR IN PART TO

AS-i Wizard in STEP 7-Micro/WIN V3.2 Service Pack 1 (V3.2.1)

AS-i Wizard

New !

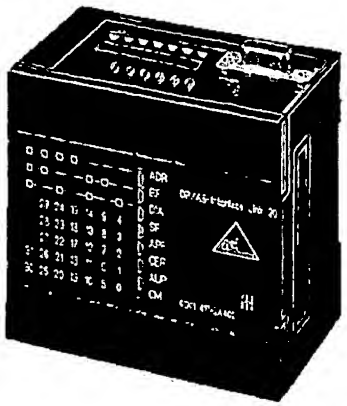


SIEMENS

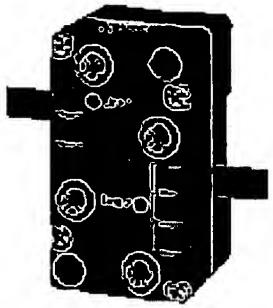
AS-i Wizard

AS-i for opportunities in lowest levels of automation

- AS-i = Actuator Sensory-interface
- AS-i systems consist of a master(CP243-2) and slave nodes that transmit simple binary signals to the master
- A dominant competitor in binary networks is Allen-Bradley's DeviceNet



AS-i CP 243-2 (Master)

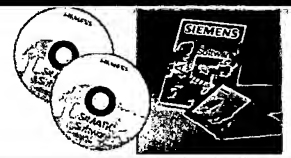


AS-i Slave

AS-i Network Cable

SIEMENS

AS-i Wizard New !

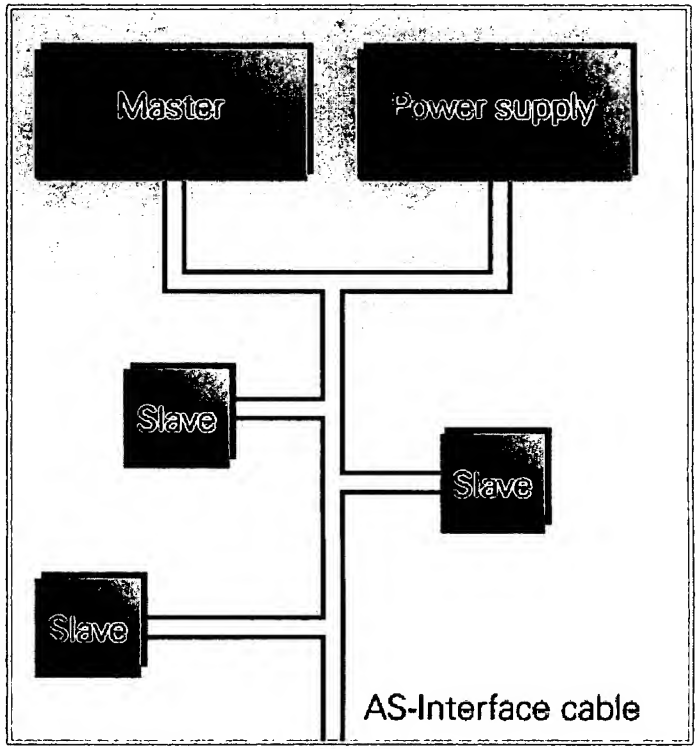


AS-i Wizard

Minimum configuration of an AS-i Network:

- One Master (CP243-2)
- Power Supply
- Slave(s)

- One control module (master) in the AS-Interface network which polls the data of the other nodes (slaves) at precisely defined intervals.
- Simple two-wire cables without shielding or PE conductor are used to carry both data and the auxiliary power for the sensors simultaneously.



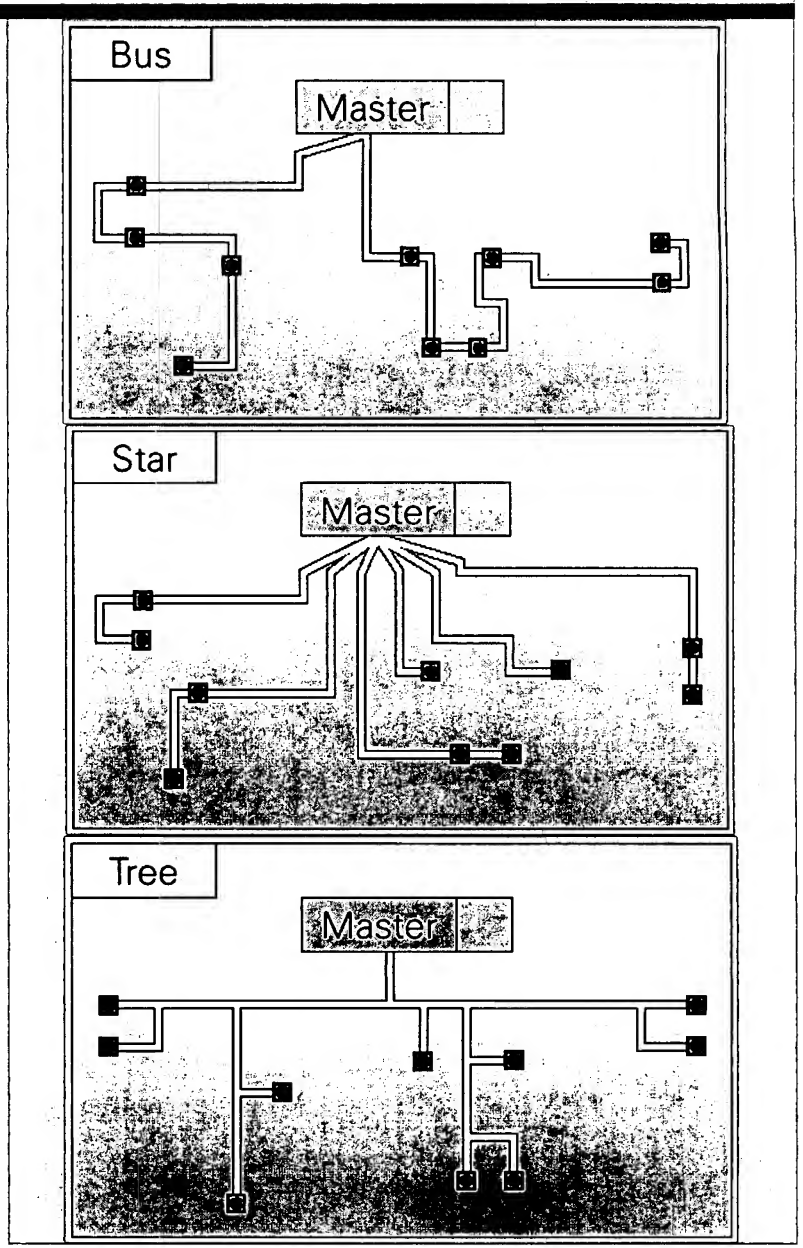
SIEMENS

AS-i Wizard New !

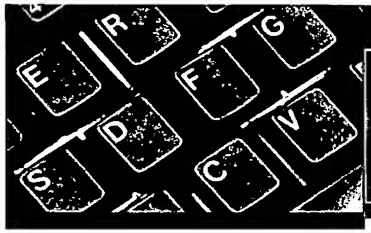
AS-i Wizard

Bus, Star, or Tree Network Configurations are possible

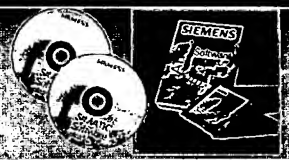
- The AS-Interface functions without any problem with standard components up to a length of 500 m – without repeaters or extenders up to 100 meters.
- Consult the AS-i documentation for additional details on network configurations, use of repeaters, etc.



SIEMENS



AS-i Wizard New !



AS-i Wizard

Without an AS-i wizard

- For S7-200 to work with AS-i, PLC programming for CP243-2 is required
- Programmer's logic must build and maintain an image register
- PLC logic to coordinate reads & writes from CP to PLC
- Tedious and error-prone
- Required to know PLC details and AS-i CP details

Advantages of AS-i wizard

- Wizard screens guide customer by asking for needed parameters
- When wizard finishes:
  - AS-i Sub-routine instructions are created based on wizard settings
  - AS-i Symbol tables are created
- Compares & updates configurations (online)
- Focus on utilizing the AS-i data, not debugging PLC logic
- Much easier - Reduces amount of programming complexity and time
- Reduces amount of technical expertise needed

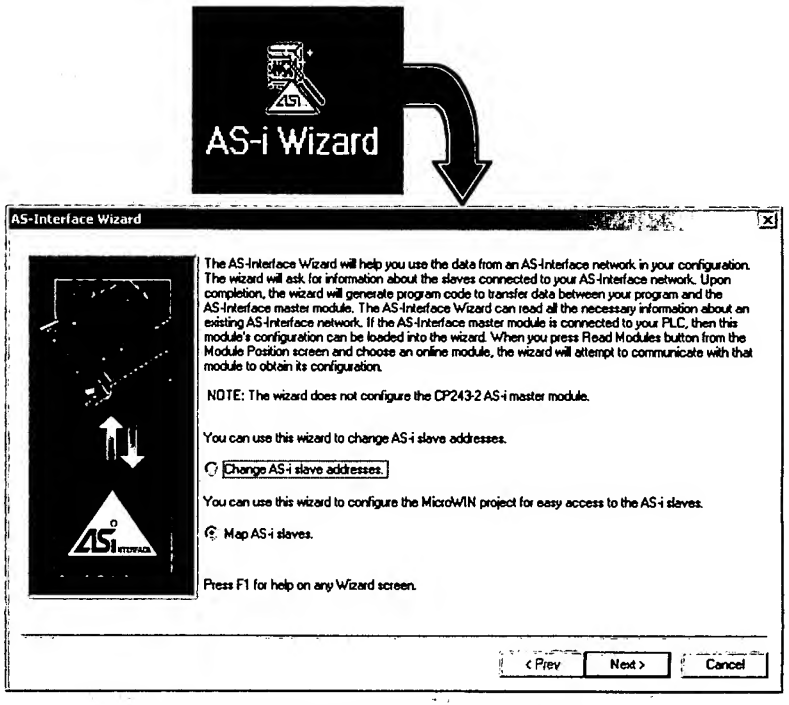
SIEMENS

AS-i Wizard New !

AS-i Wizard

Configuring an AS-i Network

- Access the wizard by clicking 'Tools' on the navigation bar, then clicking the AS-i icon.
- Consistent with other Intelligent module wizards (Context-sensitive help, etc.)
- The AS-i wizard provides a way to easily set up data transfers between the S7-200 PLC and the AS-i CP243-2 module.
- The AS-i wizard *does not* provide or replace the normal AS-i master configuration.
- When Micro/WIN is connected to an **online** AS-i network, the wizard is able to read and compare slaves on the existing network.



Introduction / Purpose

SIEMENS

AS-i Wizard New !

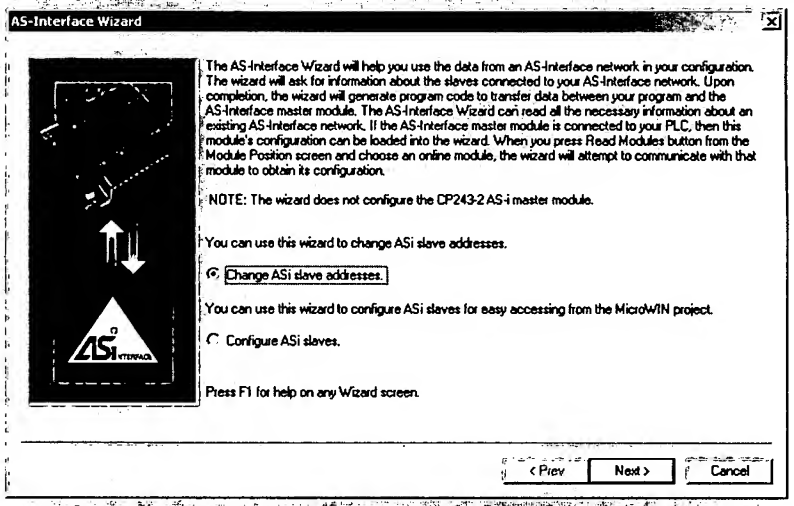
AS-i Wizard

AS-i Wizard Options

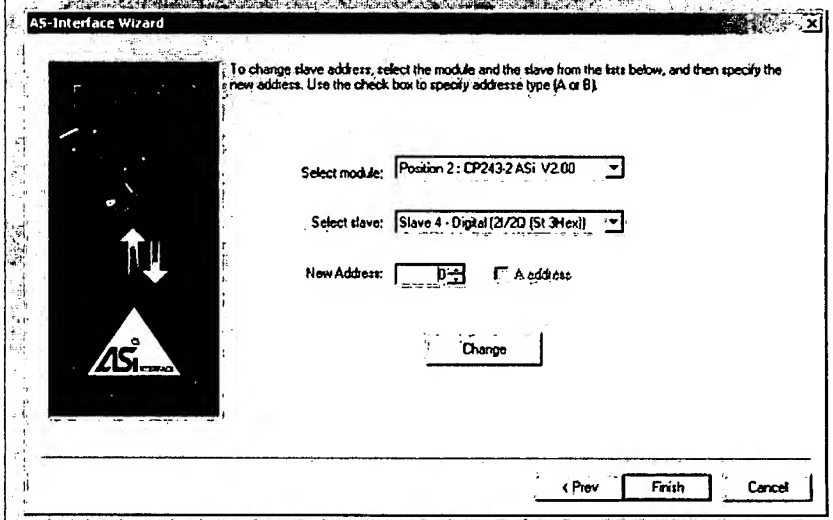
- The wizard introduction screen allows 2 options:
- Change the address of a known **online** slave.
- Map a AS-i network to the PLC

To change AS-i Slave address

- A screen is displayed to set up the changes:
- specify module position, current slave address, and new slave address
- Press the 'Change' button.
- After a slave address has been changed (after running the wizard and address is successfully changed),:then the CP243-2 reset button must be pressed (or else reset must be invoked with a hand-held programmer)

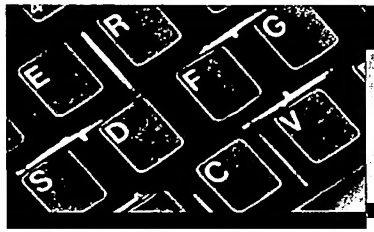


Option to change AS-i Slave Address

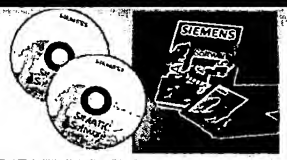


Screen to set up which AS-i slave address to change

SIEMENS



AS-i Wizard New !



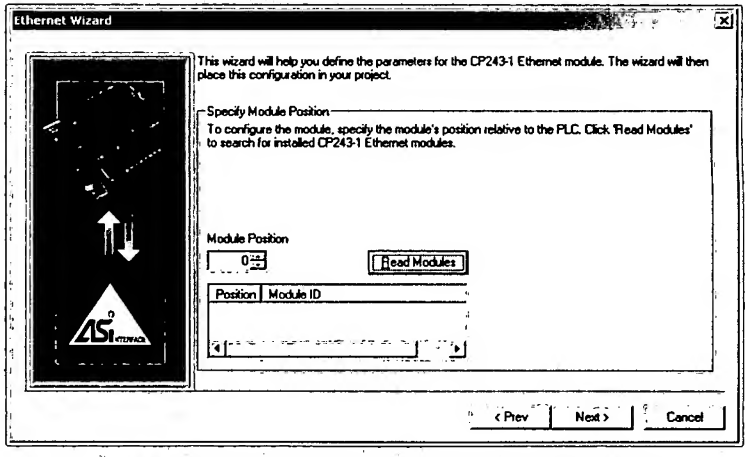
AS-i Wizard

Module Position Assignment

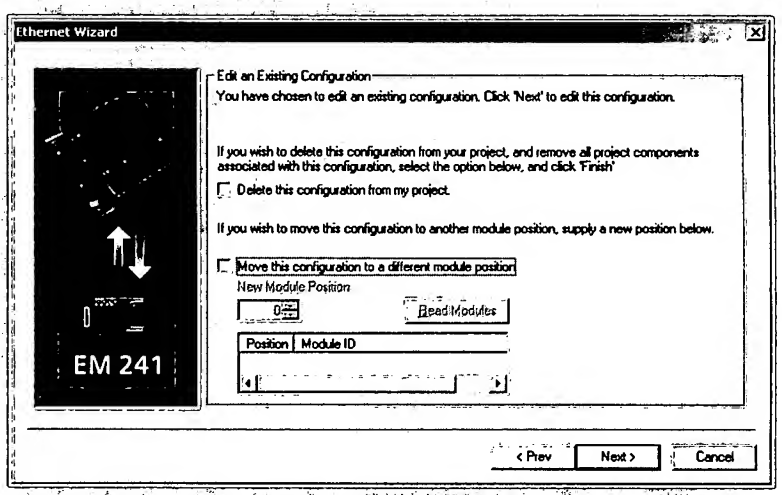
- Like all wizards, if Micro/WIN is connected to online PLC, the EM location can be found and displayed.

Re-reading the configuration

- If an AS-i configuration **already exists** in your project, when you run the AS-i wizard again….**then, on this screen:**
- The wizard detects existing configurations and asks whether you wish to modify an existing configuration or create a new configuration.
- Options will also allow programmer to **delete** or **move** the existing configuration to a different module position.



Module Position Assignment



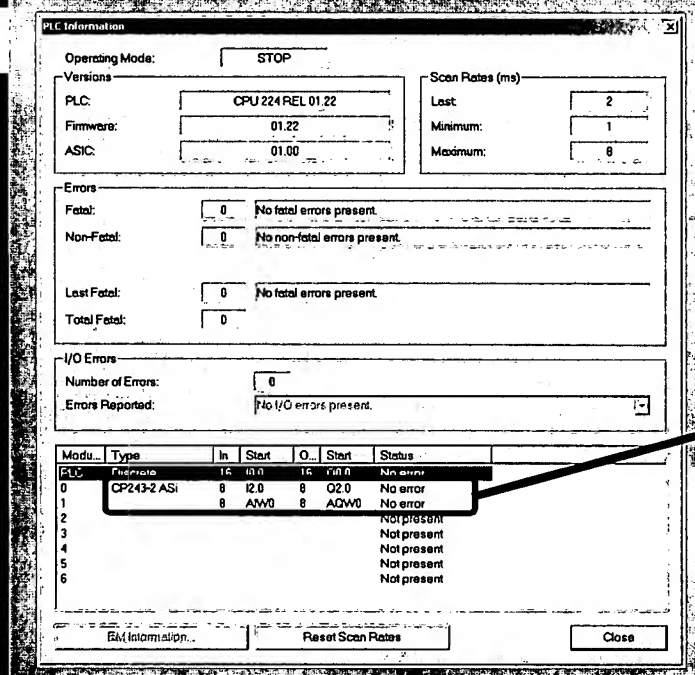
Edit Existing Configuration: Move/Delete

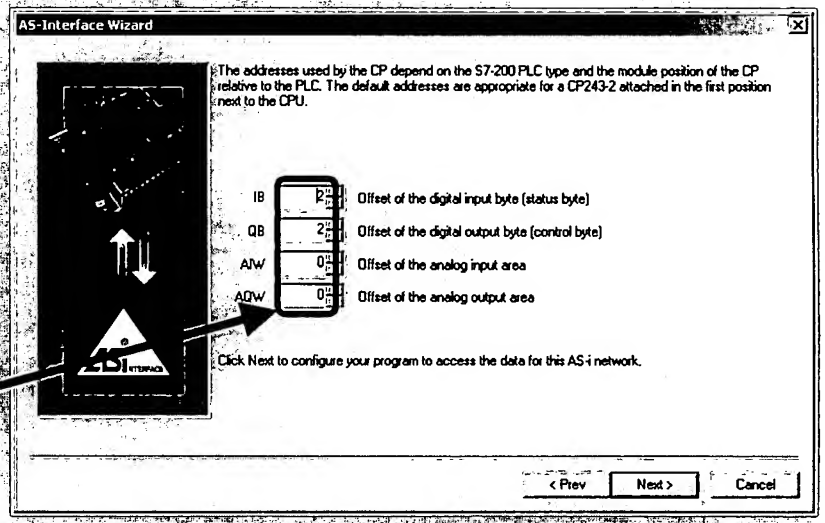
SIEMENS

AS-i Wizard

New !

AS-i Wizard





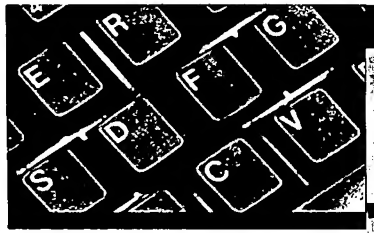
PLC Memory areas (offsets) to use

When PLC is attached to an online AS-ii network, look in PLC Info dialog: The CP243-2 to PLC mapping is shown.

The wizard assists the programmer in setting correct CP offsets:

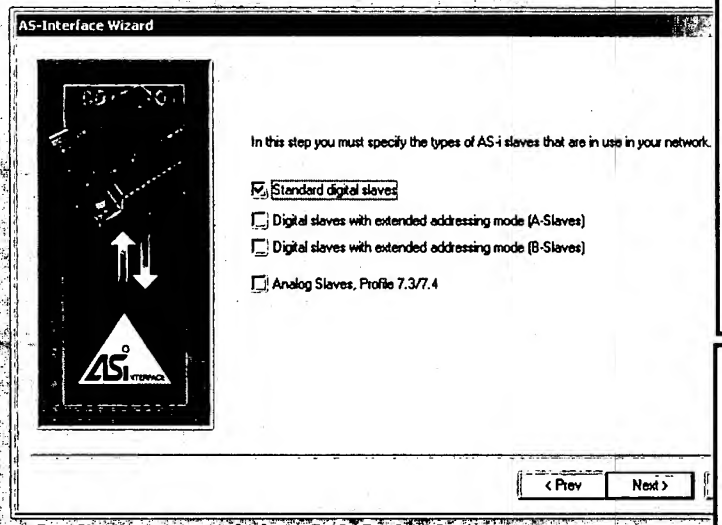
- Settings depend on: **(1)** CPU type and **(2)** AS-i CP module position.
- Offline configurations data can be optionally modified.
- Invalid configuration settings will trigger an error message from the wizard.
- For **online** EM configurations, offsets are automatically fixed and disabled (grayed out).

SIEMENS



AS-i Wizard

AS-i Wizard New !



Select type(s) of AS-I slaves to configure on your network

31 maximum slaves are allowed for:

- **Standard digital slaves**
- **Digital Slaves w/ Extended Addr. A**
- **Analog Slaves (Profile 7.3/7.4)**

These 3 slave types share the same 31 slave addresses. Therefore, **NO DUPLICATE** addresses are allowed among the 3 types.

31 maximum slaves are allowed for:

- **Digital Slaves w/ Extended Addr. B**

The Ext B slaves **CANNOT OVERLAP** any slave types **except with Digital Extended A type.**

The purpose of Extended slaves (A&B) is to allow a **maximum of 62 slave nodes** on one network.

Select the type of AS-i slaves on your network

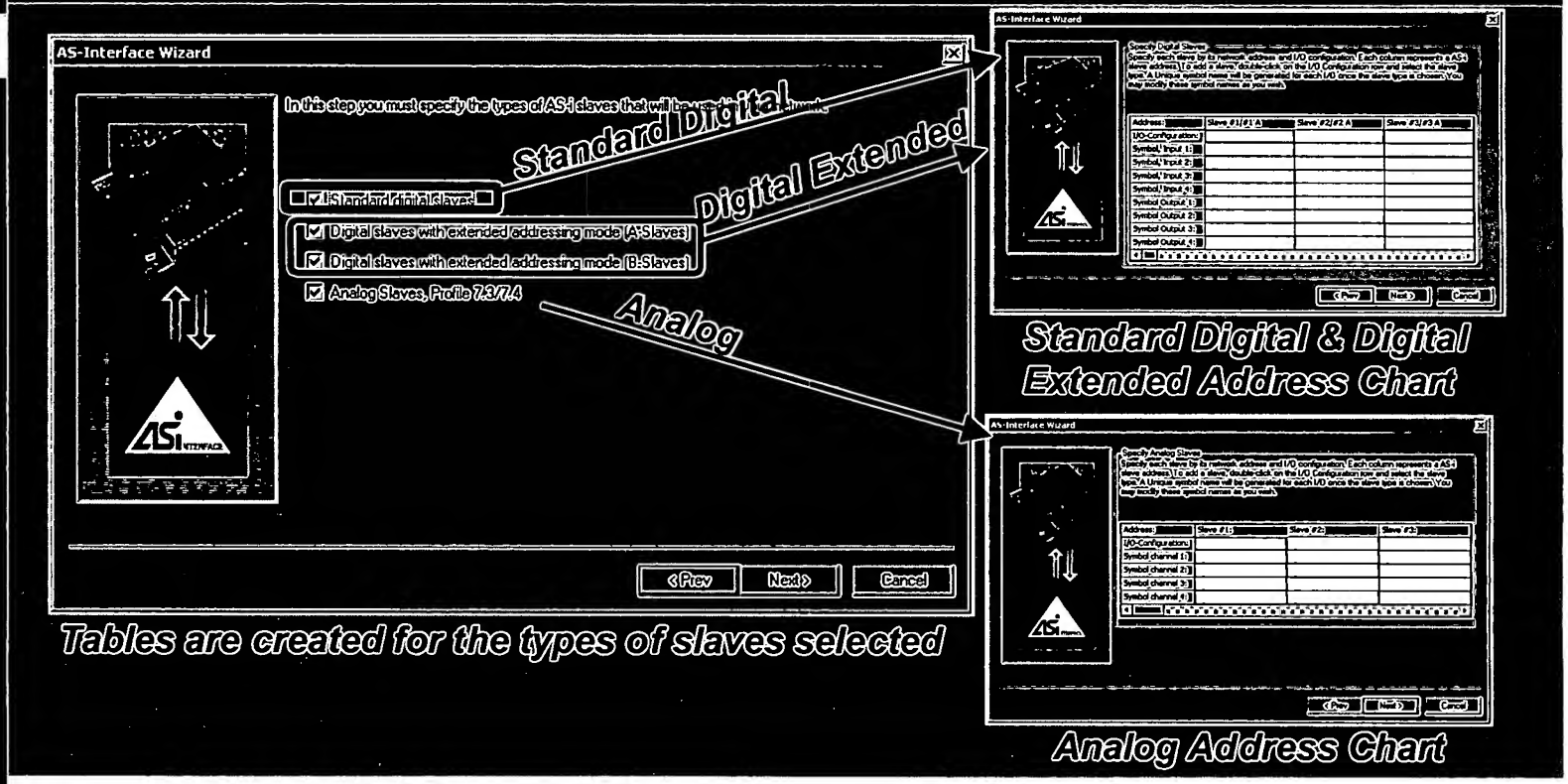
- Different slave types have unique ID codes embedded in the data frames.
- Subsequent wizard screens depend on selections made on this screen.
- For **online** EM configurations, module types are automatically set, and the checkboxes are grayed out (disabled).

Overlapped AS-i slave addresses

- For instance, there cannot be a Standard Digital Slave #2 & Analog Slave #2.
- When a slave address is already used, then the slave address column is grayed out (unavailable) to slaves in other charts. -See the following slides.

SIEMENS

AS-i Wizard New!

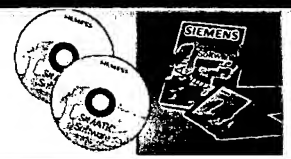


Organization of the AS-i slave types in the charts

- All of the selected digital slave types appear in one Digital chart.
- When **both** Standard Digital & Digital Ext.Address A types are selected, they both occupy the same chart columns. Example: **#1 / #1A**
- Digital Ext.Address B types follow any Standard Digital/Ext. A slaves in the digital chart. Example: **#1 - #31** followed by **#1B – #31B**

SIEMENS

AS-i Wizard New !



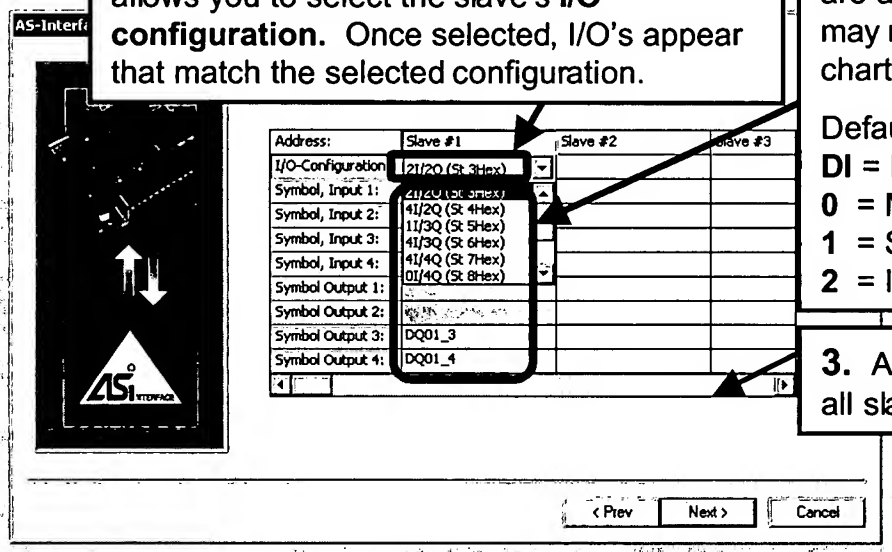
AS-i Wizard

1. Click top row for a drop-down list-box that allows you to select the slave's **I/O configuration.** Once selected, I/O's appear that match the selected configuration.

2. Symbols for individual slave I/O are assigned default symbols. You may modify the symbols in the chart.

Default Symbol Example: **DI01_2**
DI = **D**igital **I**nput,
0 = Module Position (Slot) **0**,
1 = Slave Address **1**,
2 = Input **2**

3. A **scroll bar** is used to access all slave addresses.



How a chart works.

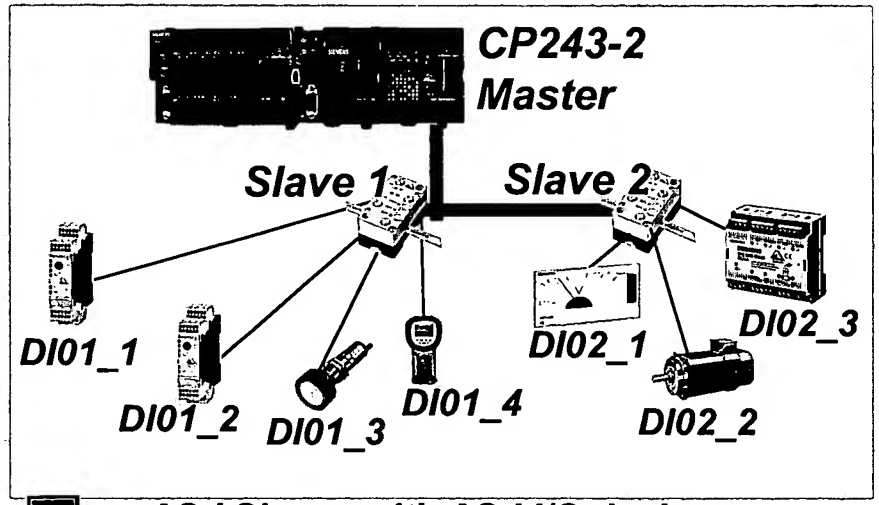
Setting up the slave data in the chart(s)

- After finishing the wizard, the slave I/O symbols appear in a symbol table.
- These symbols are for later use in the PLC program logic.
- The symbol table is re-generated when the wizard is re-run (each time 'Finish' button is pressed) for the same AS-i CP243-2 network configuration.

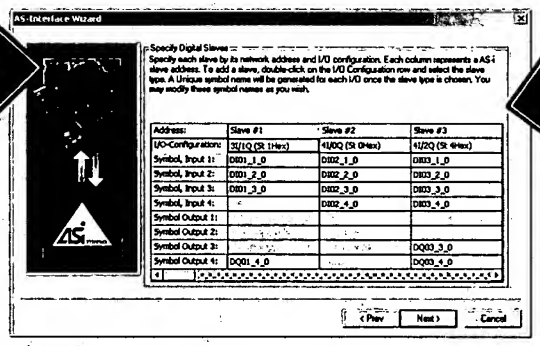
SIEMENS

AS-i Wizard New !

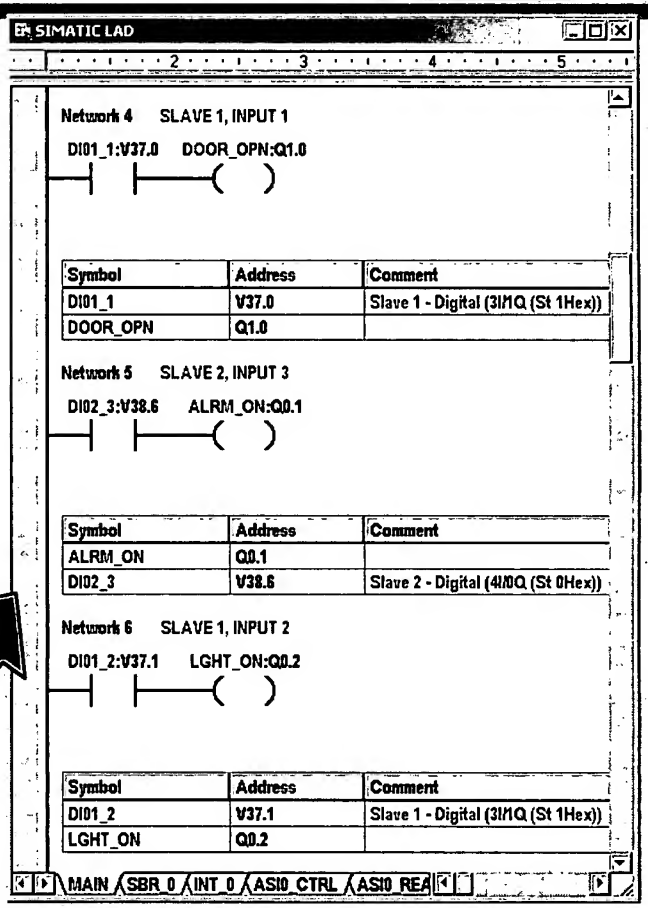
AS-i Wizard



AS-i Slaves with AS-i I/O devices



- *Chart inputs match I/O*
- *Finishing wizard generates AS-i symbols*



In the PLC logic use:
(1) The ASIx_CTRL instruction
(2) ASI I/O symbols as needed

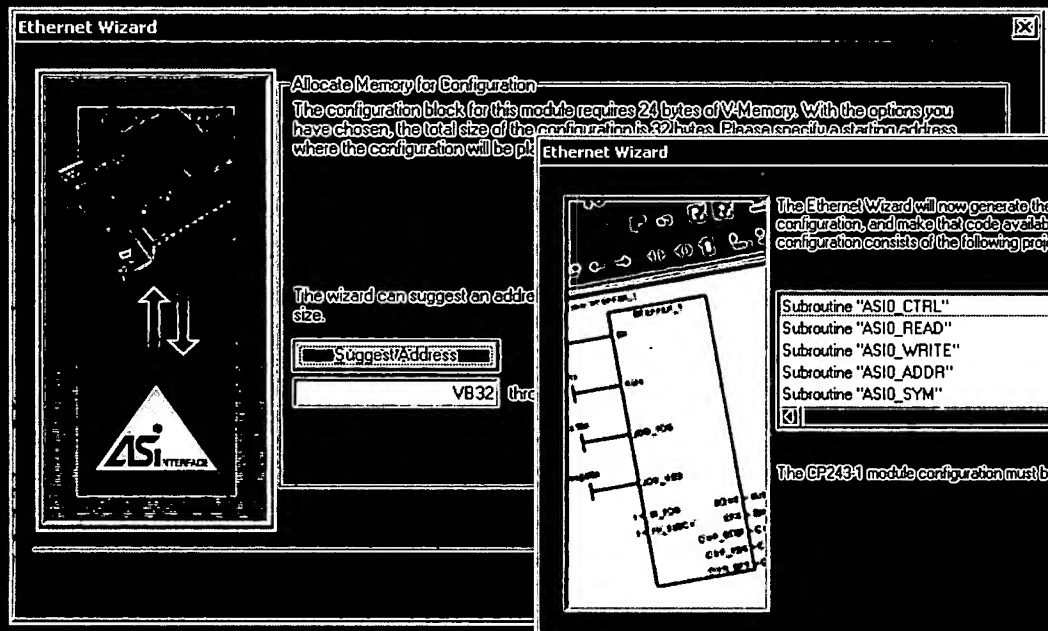
Chart symbols link PLC logic to the actual AS-i slaves

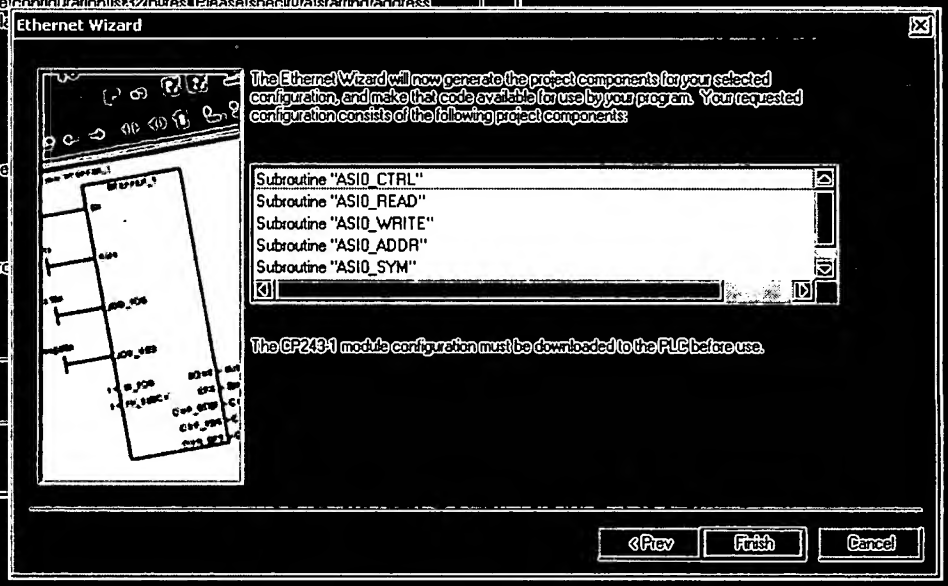
- Simply use wizard generated symbols and SBR's in your PLC logic
- The ASIx_CTRL instruction causes constant updating from AS-i CP243-2

SIEMENS

AS-i Wizard

AS-i Wizard New!!

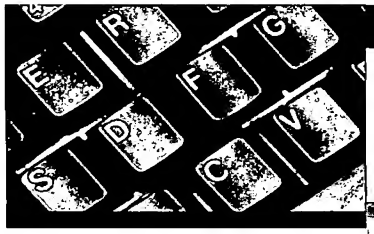




Final Wizard Screens

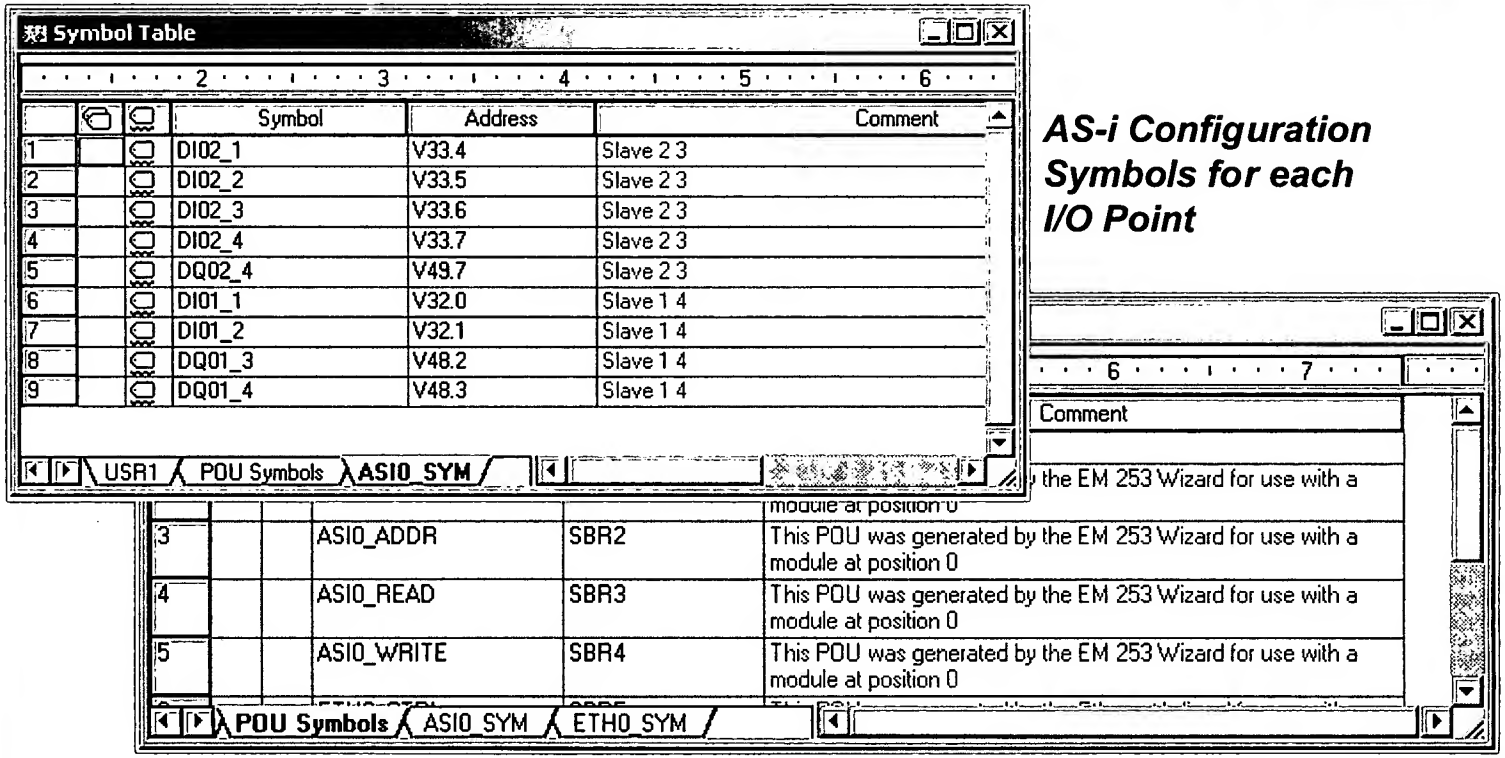
- Memory Allocation screen suggests available project space.
- The summary screen shows all project additions (SRB's, DB, Symbols).
- The summary screen gives advice about how to use the new additions.

SIEMENS



AS-i Wizard

AS-i Wizard New!



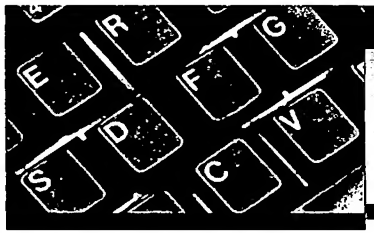
AS-i Configuration Symbols for each I/O Point

Symbols for each AS-i POU

Project is Appended By the Wizard (when 'Finish' is pressed)

- Creates a symbol table of I/O points for all the configured AS-i slaves.
- Adds the new AS-i POU's (SBR's) to the System POU Symbol table
- Use symbols from the AS-i symbol table in creating the PLC program logic
- Using the wizard (re-running the entire wizard) to change an existing configuration causes the AS-i symbol table to be re-generated (overwritten).

SIEMENS



AS-i Wizard New !

AS-i Wizard

Instructions Added to Tree

- New AS-i subroutines are added to the project's instruction tree
- **ASIx_CTRL** Instruction is used to copy slave data between the AS-i CP module and the PLC (*to be called every scan*).
- The read and write instructions are **for AS-i experts**: *they require you to know the bank number of the data**
- **ASIx_READ** Instruction is used to read bank data from the CP module (from CP to V memory).
- **ASIx_WRITE** Instruction is used to write bank data to the CP (from V memory to CP)

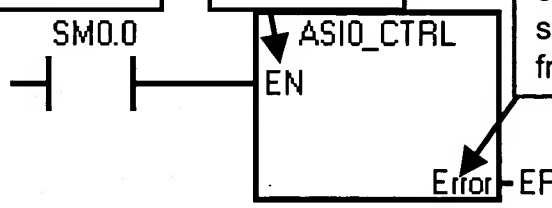
SIEMENS

* Find details about AS-i in the CP243-2 AS-i manual.

SM0.0 is ON for every scan

EN: Enables Instruction

Error : This is a location of your choice. It is not a symbol generated from the wizard.



Read & Write instructions are for AS-i Experts:

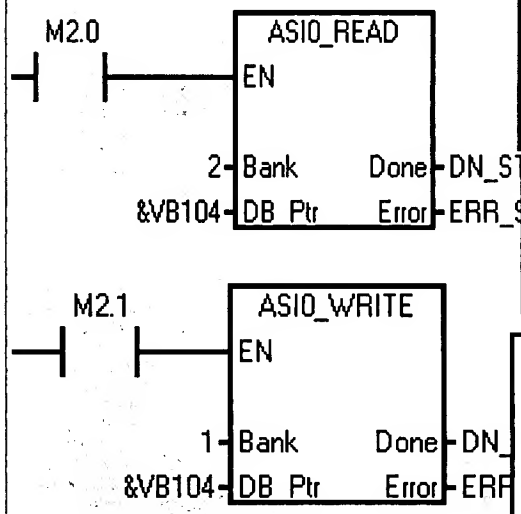
EN : Enables Instruction

Bank : AS-i bank number where data to read is located.

DB Ptr : Points to V-memory location where the AS-i data will be copied.

Done : Bit indicates when Instruction has completed.

Error –A location of your choice. A 'wizard' symbol is not generated for this error condition.

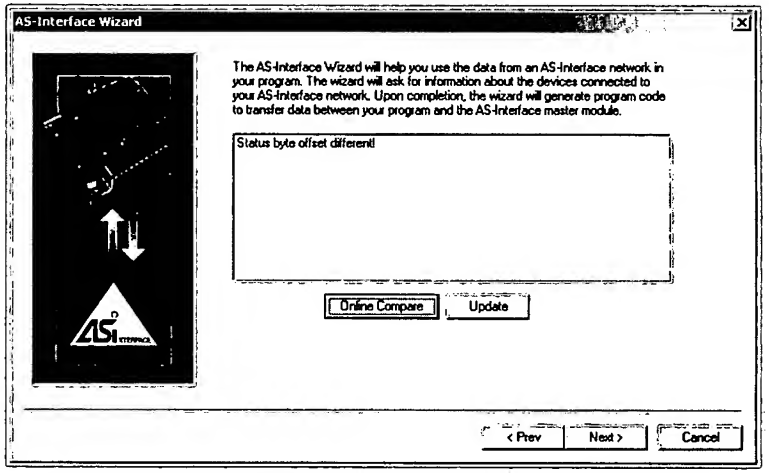


AS-i Wizard New !

AS-i Wizard

Compares the Offline& Online Configuration

- Option to compare an online PLC configuration with the offline Micro/WIN wizard configuration
- After comparing, the Micro/WIN configuration can be updated to match the online configuration
- Selecting 'Update' will:
  - **Add** missing slaves to your configuration.
  - **Replace** (write over) any offline slaves that exist with the same address(es).



Compares Offline to Online / Updates

SIEMENS

AS-i Wizard New!!

Without an AS-i wizard

- Configuring S7200-to-AS-i interfaces requires complex AS-i knowledge including program of the control/status bits, handling image registers, etc.

With an AS-i wizard

- Even beginning programmers can configure AS-i connections
- Reduces time for interfacing to an AS-i network setup
- Reduces time for modifying slaves (re-run wizard, online compare, change slave address)
- Wizard generated instructions and symbols are makes programming with the AS-i data very simple and straightforward

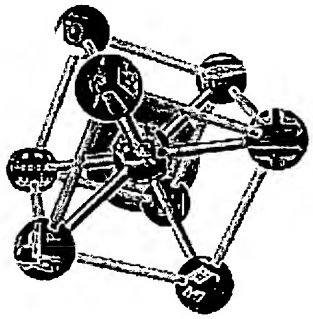
Summary

- Expands customers' usability with AS-i
- Increases S7-200 PLC capabilities in low-end a
- Allows opportunities in more application areas

SIEMENS

Document C

SIEMENS



STEP 7-Micro/WIN 32

AS-I Wizard

Test Plan

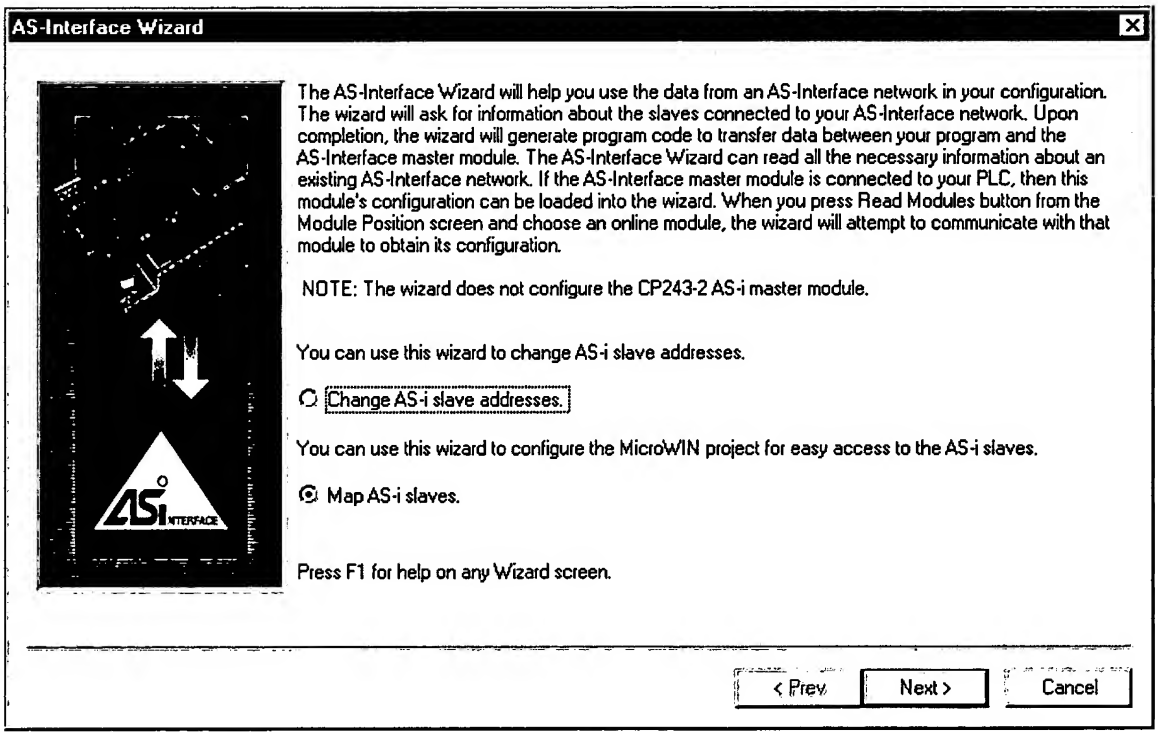
1 *Introduction*

The AS-I Wizard is a new addition to Micro/WIN for release 3.2.1. This new wizard will simplify the process of using AS-I slaves within Micro/WIN.
Not only does the wizard allow you to define symbolic names within your program, it also provides subroutines for data transfer and manipulation.
Also provided is a compare and update diagnostic tool to help in debugging your AS-I network.
This plan outlines all areas within the AS-I wizard that must be verified before product release.

2 Wizard Dialog Appearance and text

2.1 Screen 1

Like the modem wizard, the ASI wizard is two headed. From the initial screen shown below, you can either change a slave address or Map ASI slaves for use within your program.



2.1.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,

2.1.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the NEXT button is active

2.1.3 Next button

Verify that the next button takes you to the correct next window

2.1.4 Cancel button

Verify that the cancel button brings up the cancel failsafe window.
Verify that exiting the failsafe with NO leaves you in the window.
Verify that exiting the failsafe with Yes correctly closes down the wizard.

2.1.5 Window close

Verify that the window close button has the same effect as the Cancel button

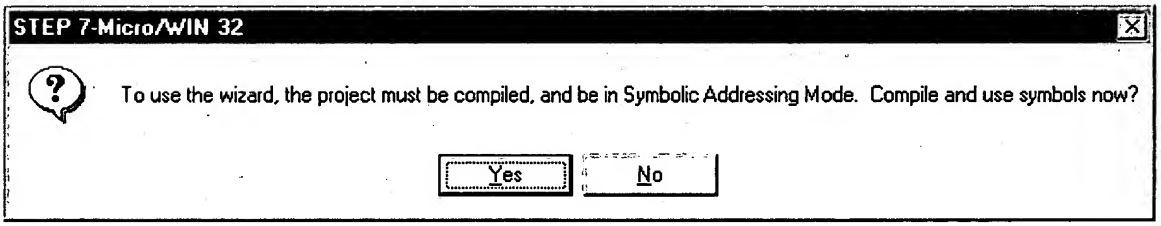
2.1.6 Change ASI slave addresses

When this radio button is active, the MAP ASI button should be inactive, these two buttons are mutually exclusive.
Verify that when this button is active, that the NEXT button takes you to the change slave address screen.

2.1.7 Map ASI slave

When this radio button is active, the Change ASI slave address should be inactive, these two buttons are mutually exclusive.
Verify that when this button is active, that the NEXT button takes you forward into the map slave window.

2.2 Screen 1A - Program compilation



2.2.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,

2.2.2 Navigation

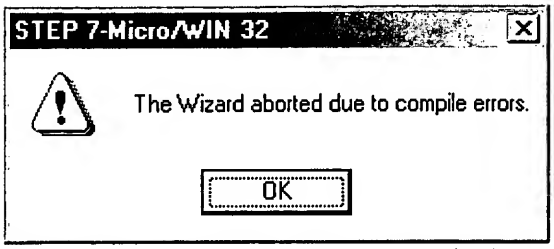
Verify tab and arrow navigation is correct.
Verify that upon entry, the YES button is active

2.2.3 NO button

Verify that the NO button cancels the wizard

2.2.4 YES button

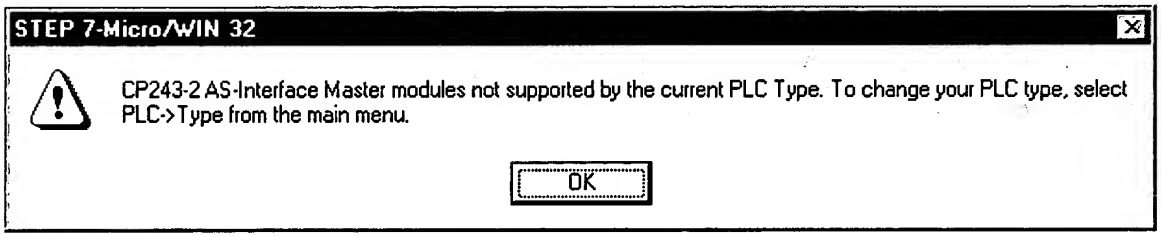
Verify that the YES button begins a program compilation
Verify that Passed compile takes you to the next screen (3)
Verify that failed compile brings up the following screen:



Verify that Pressing OK, or hitting the X dismisses this window and closes down the wizard.

2.3 Screen 1B - Invalid controller screen

Verify that the Wizard generates the following screen whenever you select a CPU type that the wizard does not support. Verify that you are allowed to proceed to screen 1 when a supported type is selected.



2.3.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,

2.3.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the OK button is active

2.3.3 OK and Close window

Verify that the OK button and the close window button shut dismiss the window and that the wizard does not start up.

2.3.4 Supported types

Verify the following CPU types are supported:

- CPU222 release 1.10 and greater
- CPU224 release 1.10 and greater
- All CPU226
- All CPU226XM

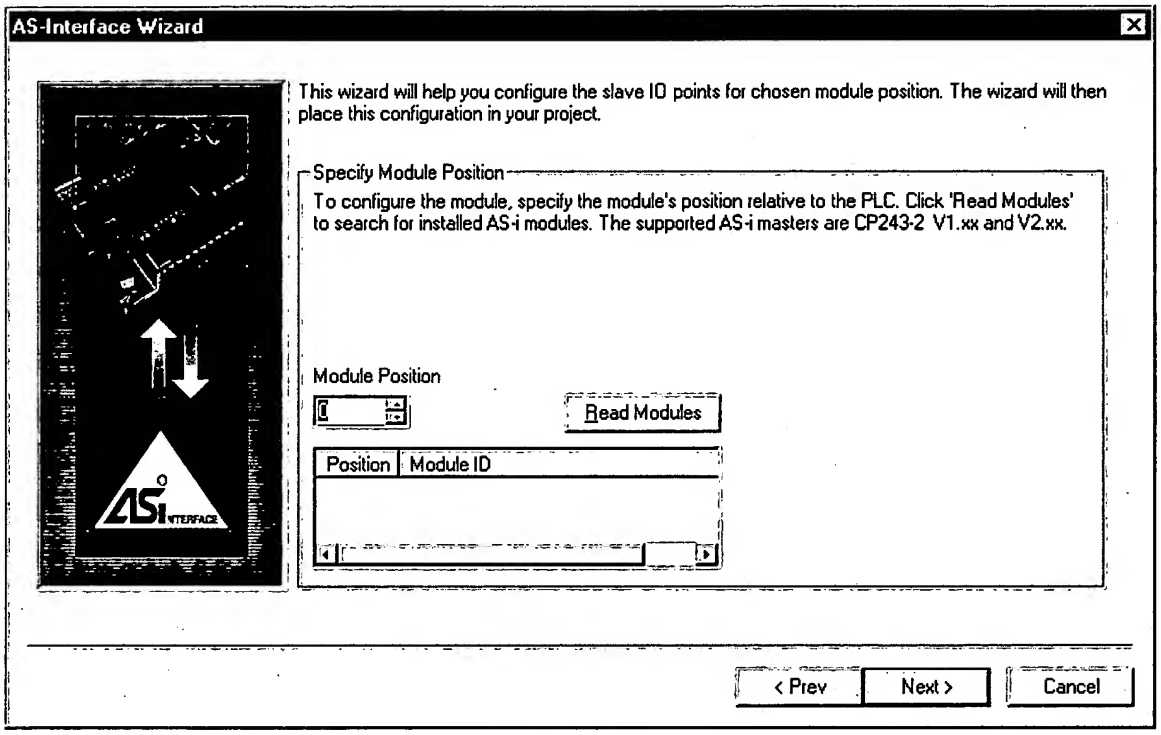
2.3.5 Unsupported types

Verify the following CPU types are unsupported:

- All Gen01
- CPU221
- CPU222 release < 1.10

CPU224 release < 1.10

2.4 Screen 2



2.4.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,

2.4.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the NEXT button is active

2.4.3 Previous button

Verify that the previous button takes you back to screen 2

2.4.4 Next button

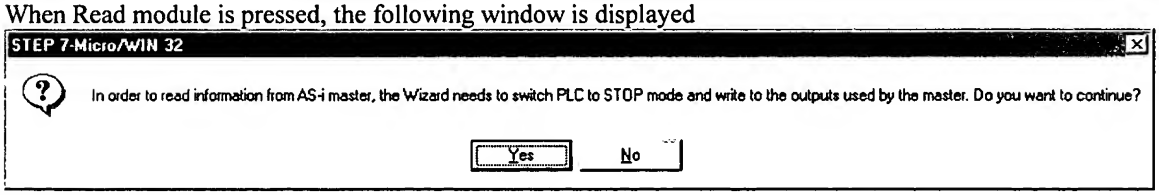
Verify that the next button takes you to the next screen

2.4.5 Cancel

Verify that the cancel button and the Window X button bring up the cancel failsafe dialog.
Verify that the wizard cancels gracefully.

2.4.6 Read modules

Attempt with no communications to the PLC
Attempt with module ready but not set
Attempt with no command acknowledgement
Attempt with error conditions returned, both known and unknown error responses.
Attempt with no modules present
Attempt with modules in each valid position
Attempt with different versions of masters connected, including less than V1.xx, V1.xx, V2.xx, and greater than V2.xx
Attempt with valid and invalid ID strings in the master
When Read module is pressed, the following window is displayed

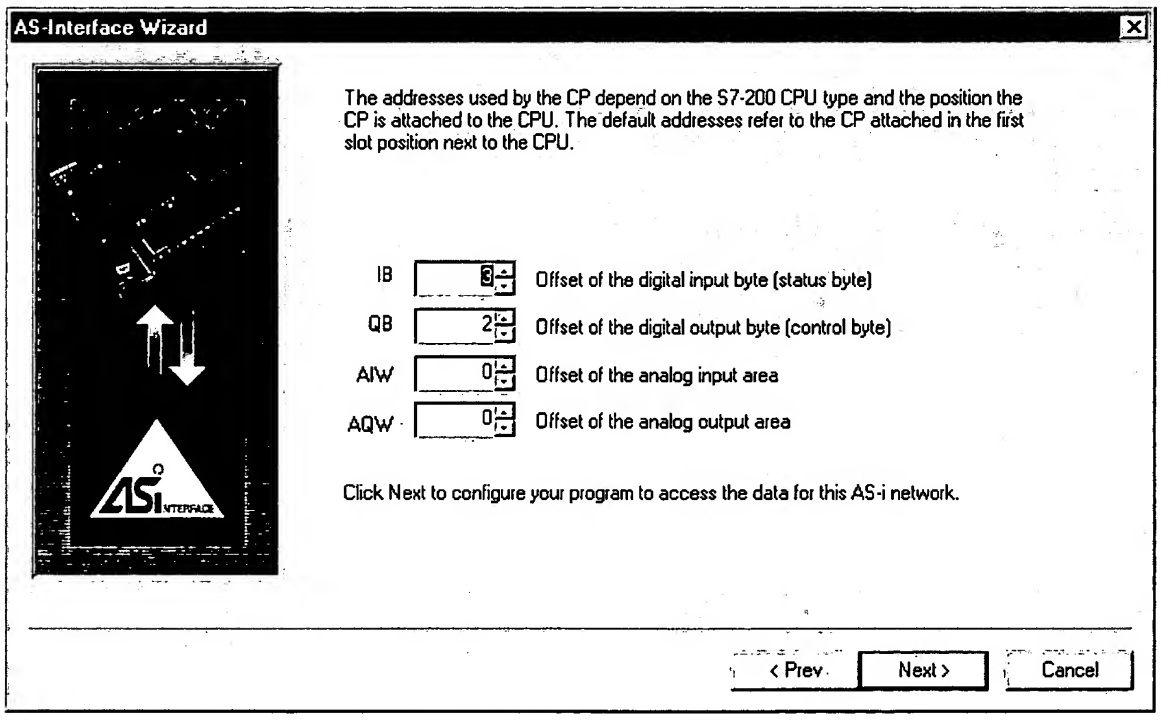


Verify navigation is correct
Verify YES is active
Verify NO cancels without placing the PLC into STOP

2.4.7 Module position spinner

Verify that the module position spinner ranges from 0 to max supported by the various controllers.
Max is 5 for most controllers (check on max for 222, think < 1.20 is 0-1)

2.5 Screen 3



2.5.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.

Verify that no truncation problems exist,

2.5.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the NEXT button is active

2.5.3 Previous button

Verify that the previous button takes you back to screen 2

2.5.4 Next button

Verify that the next button takes you to the next screen

2.5.5 Cancel

Verify that the cancel button and the Window X button bring up the cancel failsafe dialog.
Verify that the wizard cancels gracefully.

2.5.6 Disable test

Verify that when you reach this window after doing a READ modules that all spinners contain the correct value and are grayed out.

2.5.7 IB spinner

Verify the IB spinner works correctly between the ranges of 0-15

2.5.8 QB spinner

Verify the QB spinner works correctly between the ranges of 0-15

2.5.9 AIW spinner

Make sure the AIW spinner works correctly between the ranges of 0-62 in increments of 2 for controllers > CPU222. Ranges for the CPU222 are 0-16

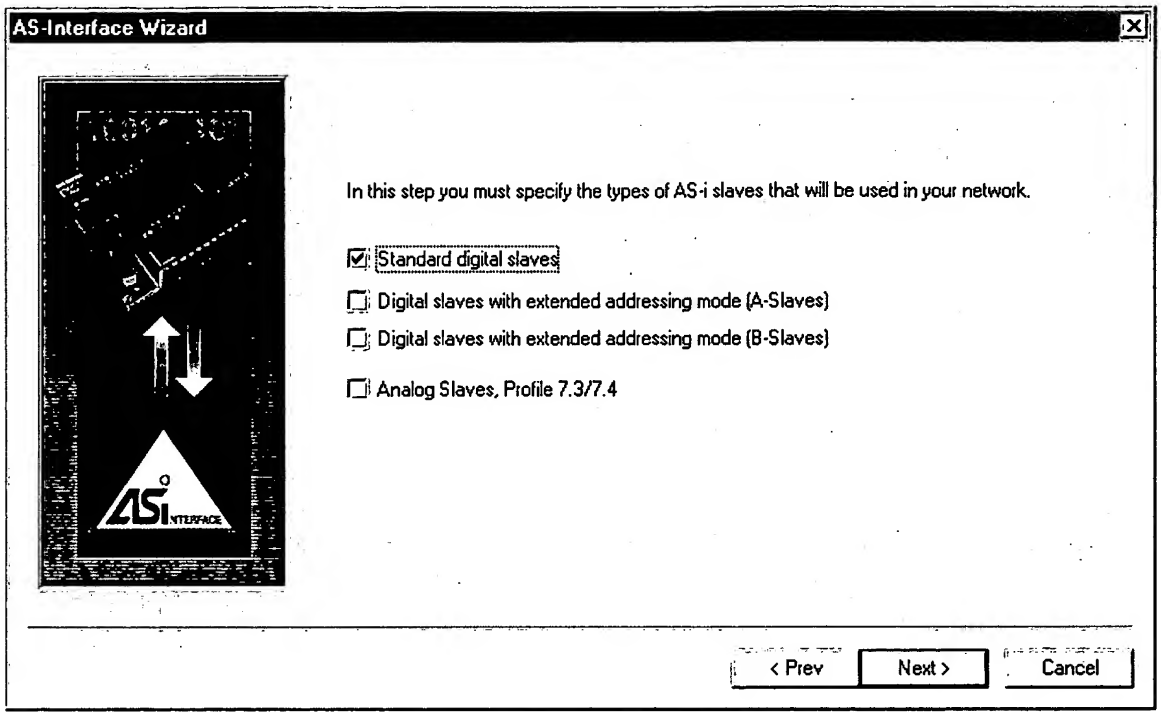
2.5.10 AQW spinner

Make sure the AQW spinner works correctly between the ranges of 0-62 in increments of 2 for controllers larger than the CPU222. Ranges for the CPU222 are 0-16

2.5.11 Read Modules fill in

Verify that when Read modules was pressed in the previous screen that the correct ranges are placed in these fields and that they are locked.
Verify for each CPU type supported in a variety of positions.

2.6 Screen 4



2.6.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,

2.6.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the NEXT button is active

2.6.3 Previous button

Verify that the previous button takes you back to screen 2

2.6.4 Next button

Verify that the next button takes you to the next screen

2.6.5 Cancel

Verify that the cancel button and the Window X button bring up the cancel failsafe dialog.
Verify that the wizard cancels gracefully.

2.6.6 Disable test

When an AS-I master is found with the READ modules function, it will be identified as unknown, V1.x or V2.x. Only the V2.x or above supports extended addressing and analog slaves. These will be grayed out of unsupported by the master.

2.6.6.1 V BETA master

Use the simulator to return a non 1.x or 2.x master. Verify that this case is handled correctly

2.6.6.2 V1.x master

Use the simulator to return a V1.x master. Verify that the Digital slaves checkbox is enabled and checked.
Verify that the remaining 3 are grayed out and unchecked.

2.6.6.3 V2.x master

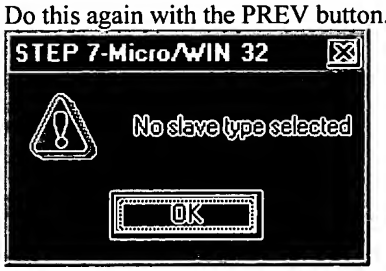
Verify that all slave boxes are enabled and checked

2.6.6.4 No read

Get to this window without performing a READ module function. Verify that all check boxes are enabled
Verify that the first is checked and the remaining 3 are unchecked.

2.6.6.5 No slave selected

Uncheck all boxes and press the NEXT button
Verify the following window is displayed.
Do this again with the PREV button.



2.6.6.6 Field locking

Whenever a slave is entered into a grid later in the wizard, the field corresponding to that slave type becomes selected and locked.
Go forward and enter a standard slave. Back up and verify that the standard slaves field is locked.
Repeat this for A slaves, B slaves, and Analog slaves.
Try different combinations.

2.7 Screen 5 - Digital Slave

This screen allows the user to enter configurations for digital slaves.
There are six variants of this screen depending on whether the user has selected extended slaves.
There is much common behavior shared between each of these variants. It needs to be checked against only one variant.

2.7.1 Shared behaviors

2.7.1.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,

2.7.1.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the NEXT button is active

2.7.1.3 Previous button

Verify that the previous button takes you back to screen 2

2.7.1.4 Next button

Verify that the next button takes you to the next screen

2.7.1.5 Cancel

Verify that the cancel button and the Window X button bring up the cancel failsafe dialog.
Verify that the wizard cancels gracefully.

2.7.1.6 Invalid symbol names

Enter invalid names within a wide range of cells, verify that they are displayed in red when you leave the cell.
Verify that you are not allowed to leave this window if invalid symbol names exist.

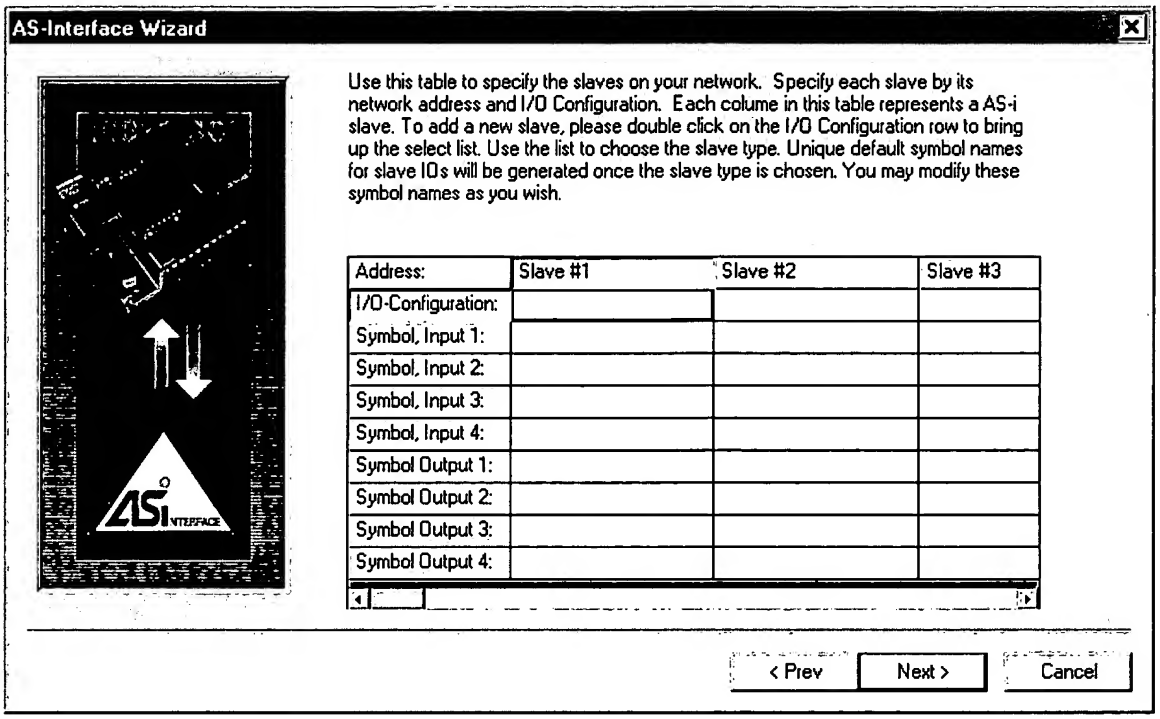
2.7.1.7 Duplicate symbol names

Enter duplicate names within a wide range of cell, verify that they are displayed in with a green squiggly line when you leave the cell.
Verify that you are not allowed to leave this window if duplicate symbol names exist.
Be sure to verify duplicates symbols are caught when duplicates exist in the following places:
Symbols within the current grid
User symbol table
Other wizard protected symbol tables
Symbol tables of AS-I modules in different positions
Symbols defined in the analog grid
Verify that all instances of the duplication are caught. For example, define a duplicate symbol in several cells and verify that they are caught.

2.7.1.8 Creation of unique symbol names

Whenever you drop a new slave into the grid, a unique symbol name must be created. Verify that this is so by defining symbols in all the places specified in the duplicate symbol names section.
Verify that

2.7.2 Standard Slave



Slave types supported for this variant can be found in the specification.
Verify that all valid slaves for this type can be updated from the controller. This includes the special actuator and sensor types.
Verify correct behavior when an update occurs that includes Extended A and B slaves.
Verify correct behavior when an update occurs that contains analog slaves
Verify correct behavior when an update occurs that contains unknown slave types.

2.7.2.1 Read from CPU tests

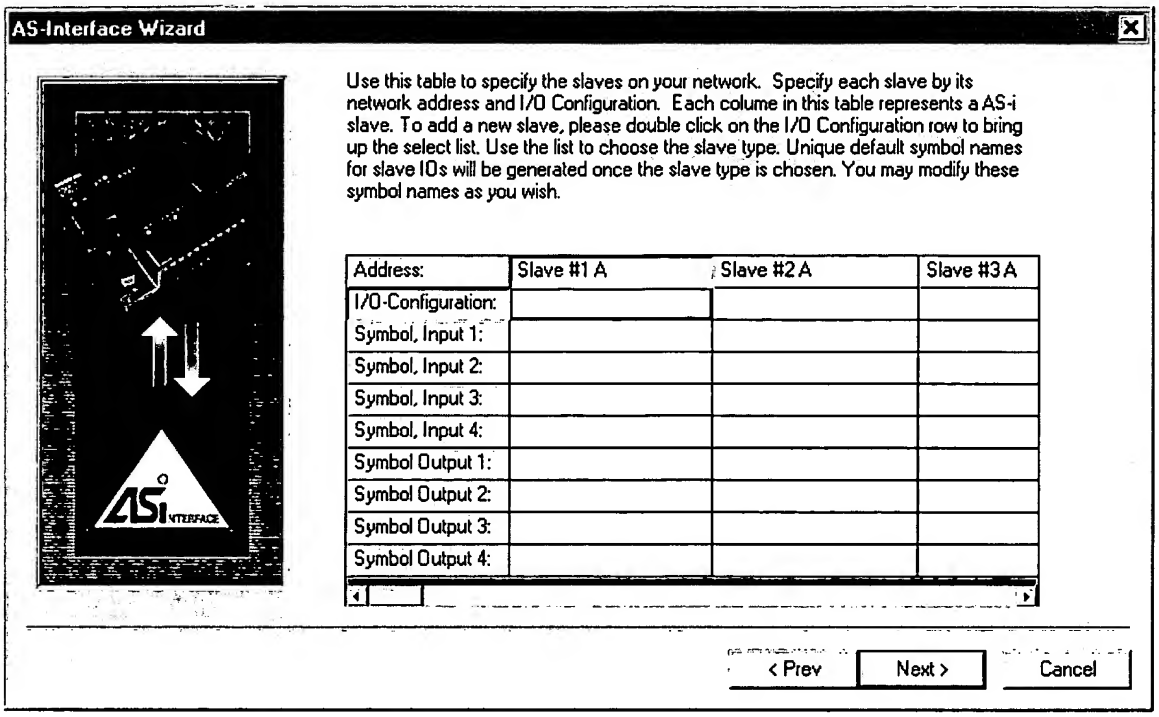
2.7.2.1.1 Valid slaves

Verify that when slaves are read from the controller that all valid slaves are recognized in the correct positions. This includes all profiles and Ids, including extended Ids.

2.7.2.1.2 Invalid slaves

Verify that invalid slave types are detected and displayed for all slave locations. Invalid slave types include:

2.7.3 Extended A slave only



Slave types supported for this variant can be found in the specification.
Verify that all valid slaves for this type can be updated from the controller. This includes the special actuator and sensor types.
Verify correct behavior when an update occurs that includes standard and Extended B slaves.
Verify correct behavior when an update occurs that contains analog slaves
Verify correct behavior when an update occurs that contains unknown slave types.

2.7.3.1 Read from CPU tests

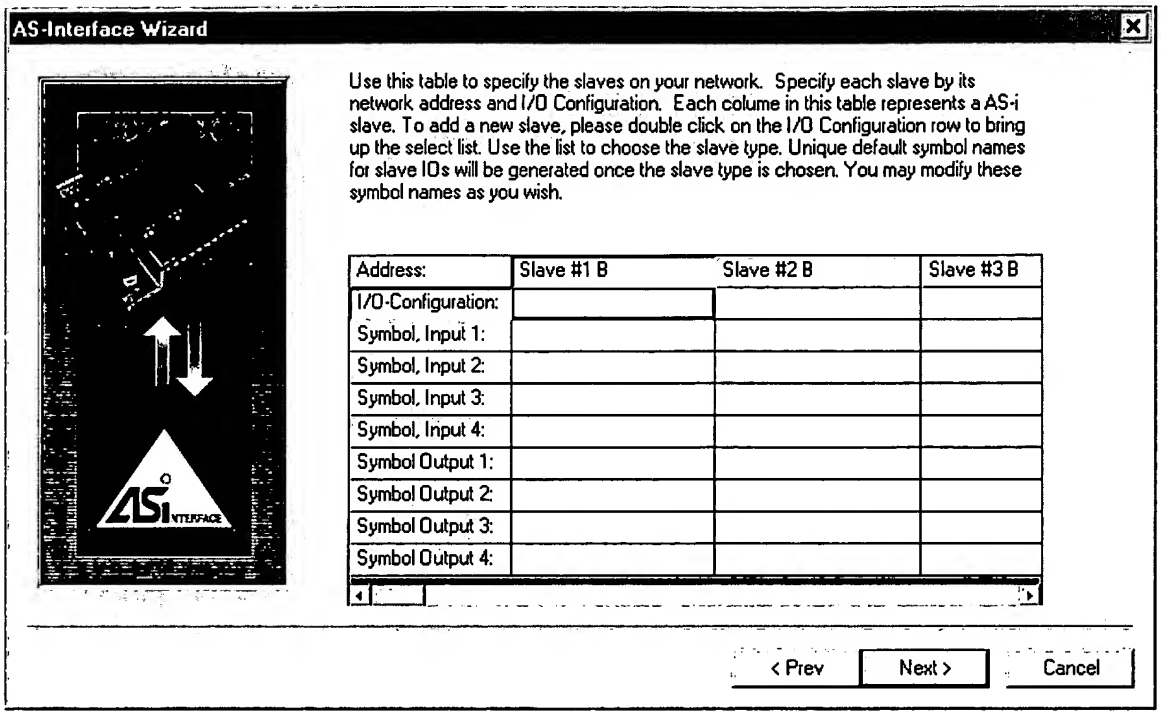
2.7.3.1.1 Valid slaves

Verify that when slaves are read from the controller that all valid slaves are recognized in the correct positions. This includes all profiles and Ids, including extended Ids.

2.7.3.1.2 Invalid slaves

Verify that invalid slave types are detected and displayed for all slave locations. Invalid slave types include:
Invalid standard slaves
Invalid A slaves

2.7.4 Extended B slave only



Slave types supported for this variant can be found in the specification.
Verify that all valid slaves for this type can be updated from the controller. This includes the special actuator and sensor types.
Verify correct behavior when an update occurs that includes standard and Extended A slaves.
Verify correct behavior when an update occurs that contains analog slaves
Verify correct behavior when an update occurs that contains unknown slave types.

2.7.4.1 Read from CPU tests

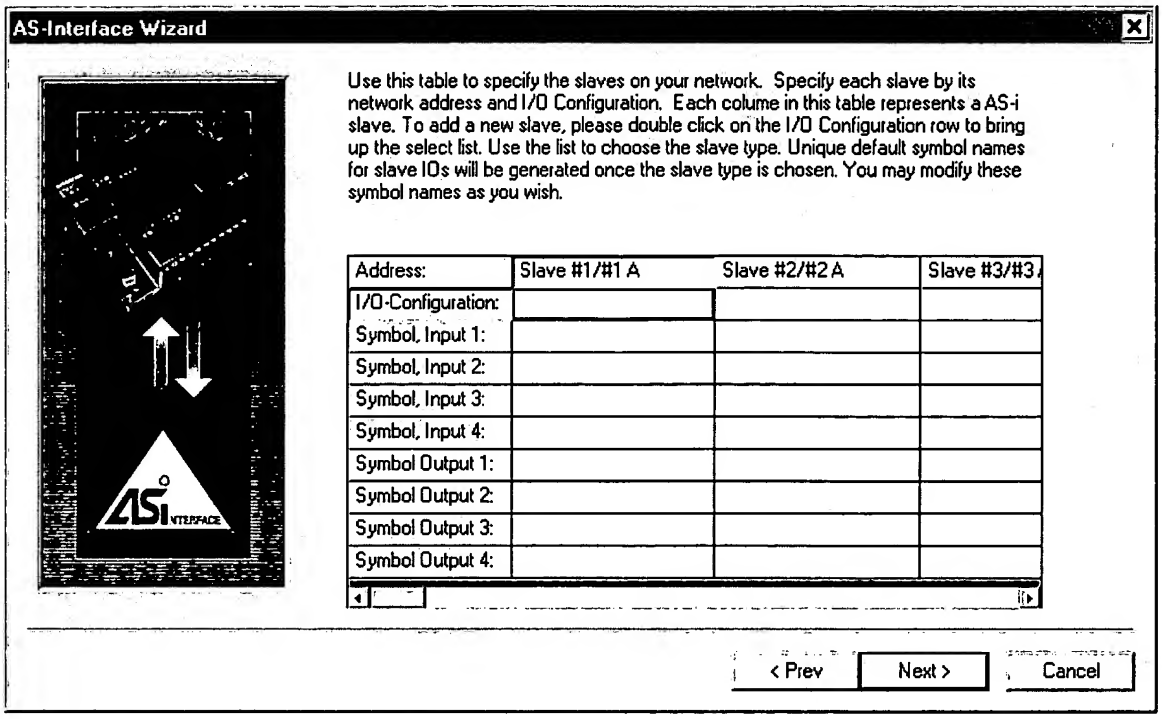
2.7.4.1.1 Valid slaves

Verify that when slaves are read from the controller that all valid slaves are recognized in the correct positions. This includes all profiles and Ids, including extended Ids.

2.7.4.1.2 Invalid slaves

Verify that invalid slave types are detected and displayed for all slave locations. Invalid slave types include:
Any standard slave
Any Analog slave
Invalid B slaves

2.7.5 Standard and extended A slave



Slave types supported for this variant can be found in the specification.
Verify that all valid slaves for this type can be updated from the controller. This includes the special actuator and sensor types.
Verify correct behavior when an update occurs that includes Extended B slaves.
Verify correct behavior when an update occurs that contains analog slaves
Verify correct behavior when an update occurs that contains unknown slave types.

2.7.5.1 Read from CPU tests

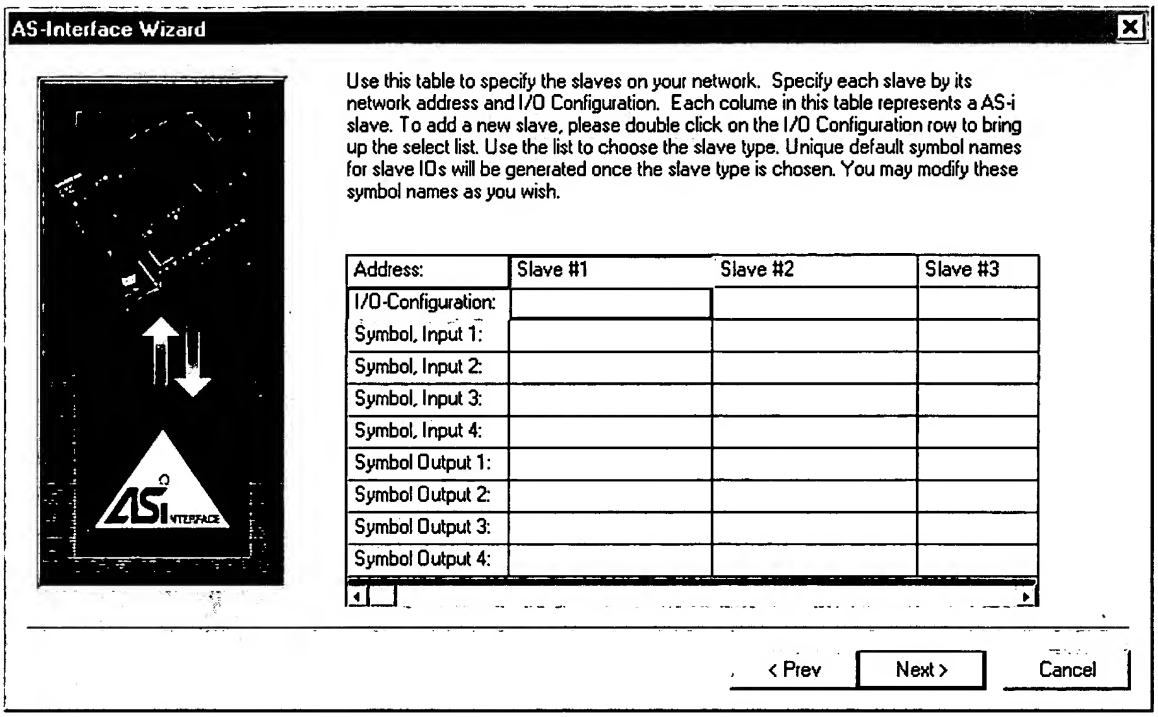
2.7.5.1.1 Valid slaves

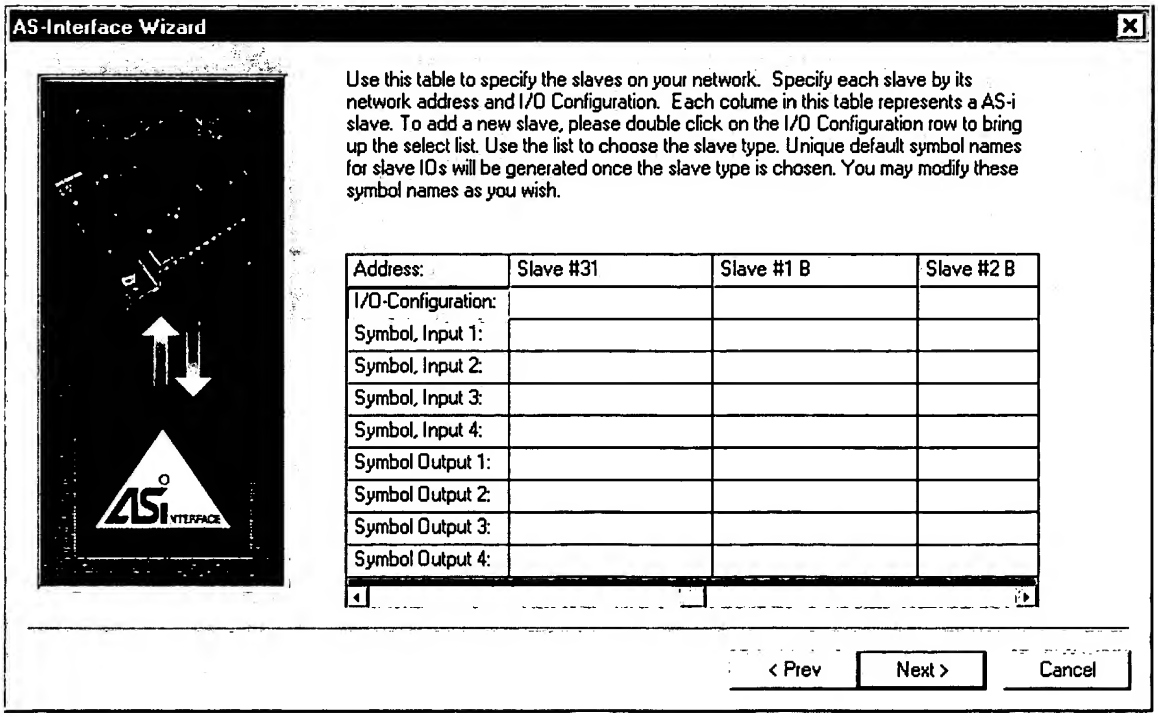
Verify that when slaves are read from the controller that all valid slaves are recognized in the correct positions. This includes all profiles and Ids, including extended Ids.

2.7.5.1.2 Invalid slaves

Verify that invalid slave types are detected and displayed for all slave locations. Invalid slave types include:
Any standard slave
Any Analog slave
Invalid B slaves

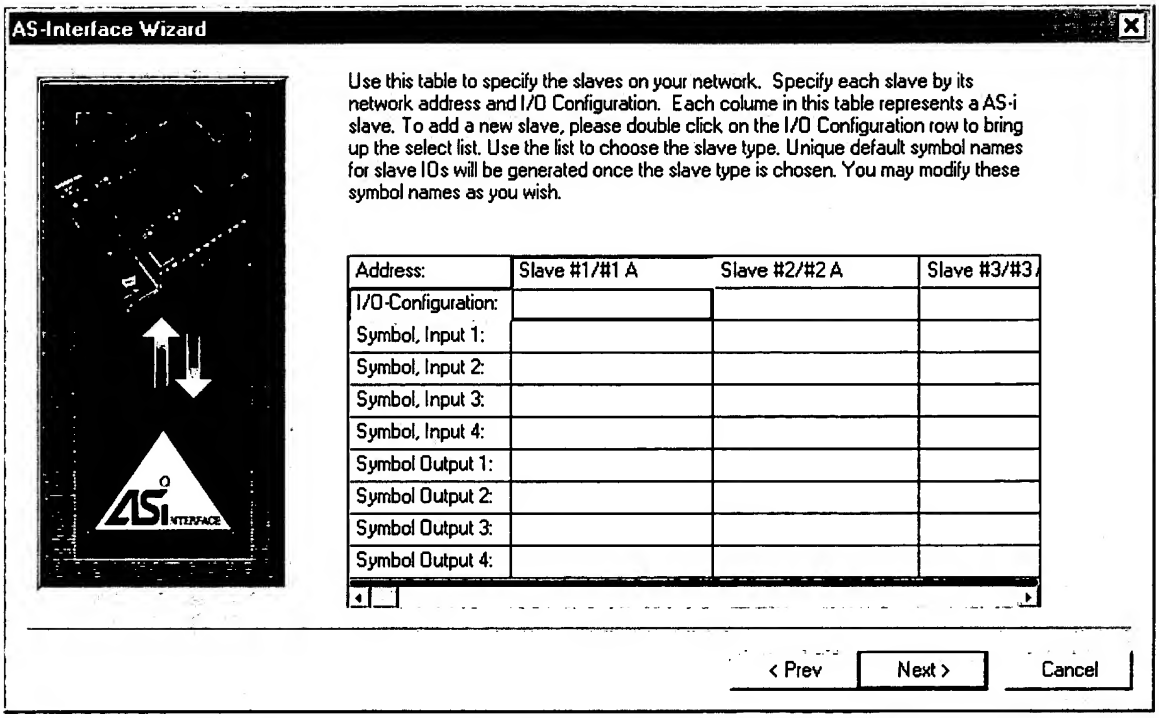
2.7.6 Standard and extended B slave

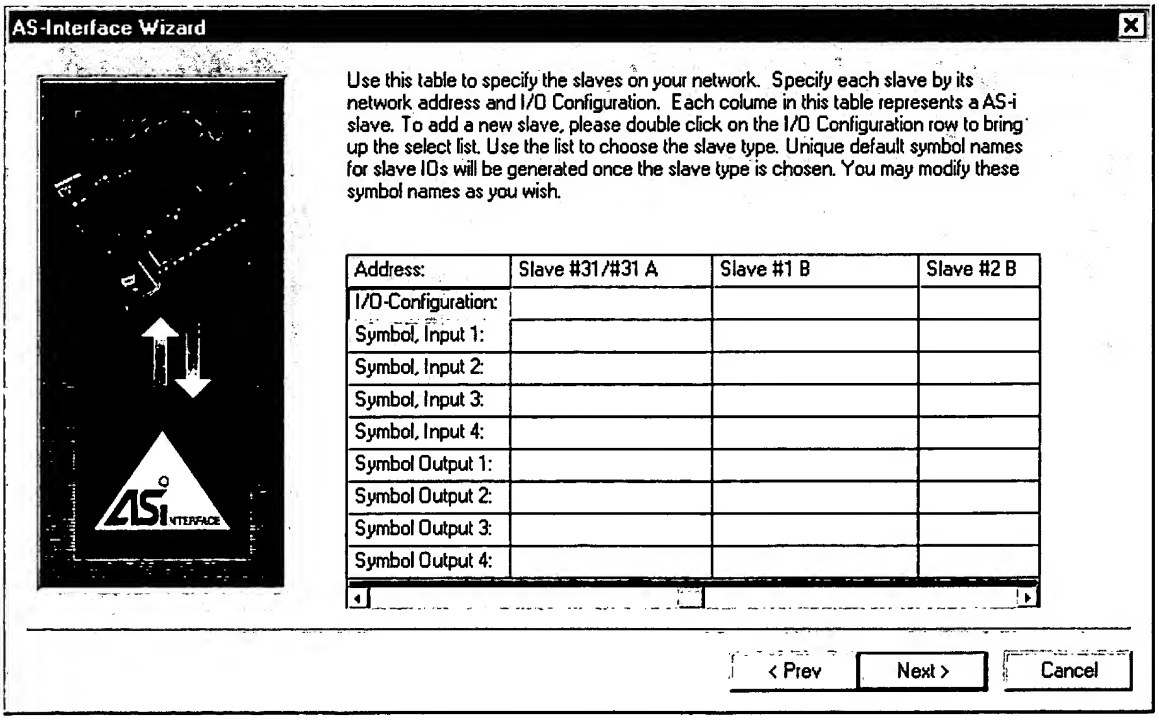




Slave types supported for this variant can be found in the specification.
Verify that all valid slaves for this type can be updated from the controller. This includes the special actuator and sensor types.
Verify correct behavior when an update occurs that includes Extended A slaves.
Verify correct behavior when an update occurs that contains analog slaves
Verify correct behavior when an update occurs that contains unknown slave types.

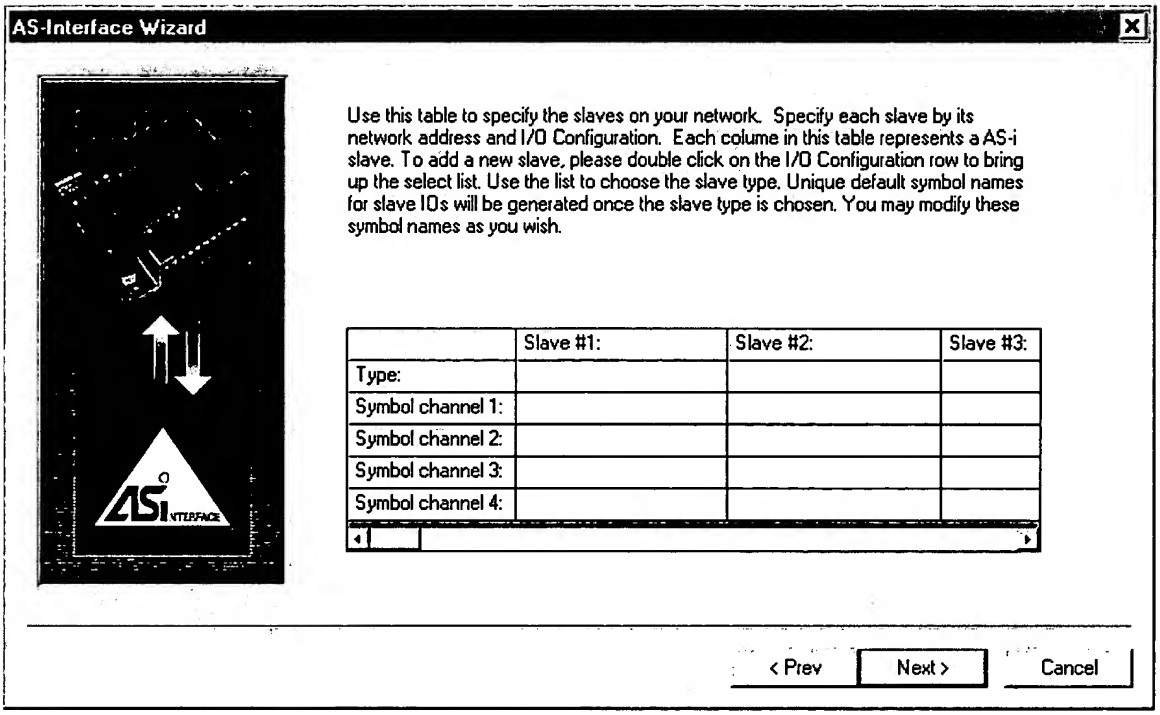
2.7.7 Standard with extended A and B slaves





Slave types supported for this variant can be found in the specification.
Verify that all valid slaves for this type can be updated from the controller. This includes the special actuator and sensor types.
Verify correct behavior when an update occurs that contains analog slaves
Verify correct behavior when an update occurs that contains unknown slave types.

2.8 Screen 6 Analog Slave



2.8.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,

2.8.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the NEXT button is active

2.8.3 Previous button

Verify that the previous button takes you back to screen 2

2.8.4 Next button

Verify that the next button takes you to the next screen

2.8.5 Cancel

Verify that the cancel button and the Window X button bring up the cancel failsafe dialog.
Verify that the wizard cancels gracefully.

2.8.6 Invalid symbol names

Enter invalid names within a wide range of cells, verify that they are displayed in red when you leave the cell.

Verify that you are not allowed to leave this window if invalid symbol names exist.

2.8.7 Duplicate symbol names

Enter duplicate names within a wide range of cell, verify that they are displayed in with a green squiggly line when you leave the cell.
Verify that you are not allowed to leave this window if duplicate symbol names exist.
Be sure to verify duplicates symbols are caught when duplicates exist in the following places:
Symbols within the current grid
User symbol table
Other wizard protected symbol tables
Symbol tables of AS-I modules in different positions
Symbols defined in the analog grid
Verify that all instances of the duplication are caught. For example, define a duplicate symbol in several cells and verify that they are caught.

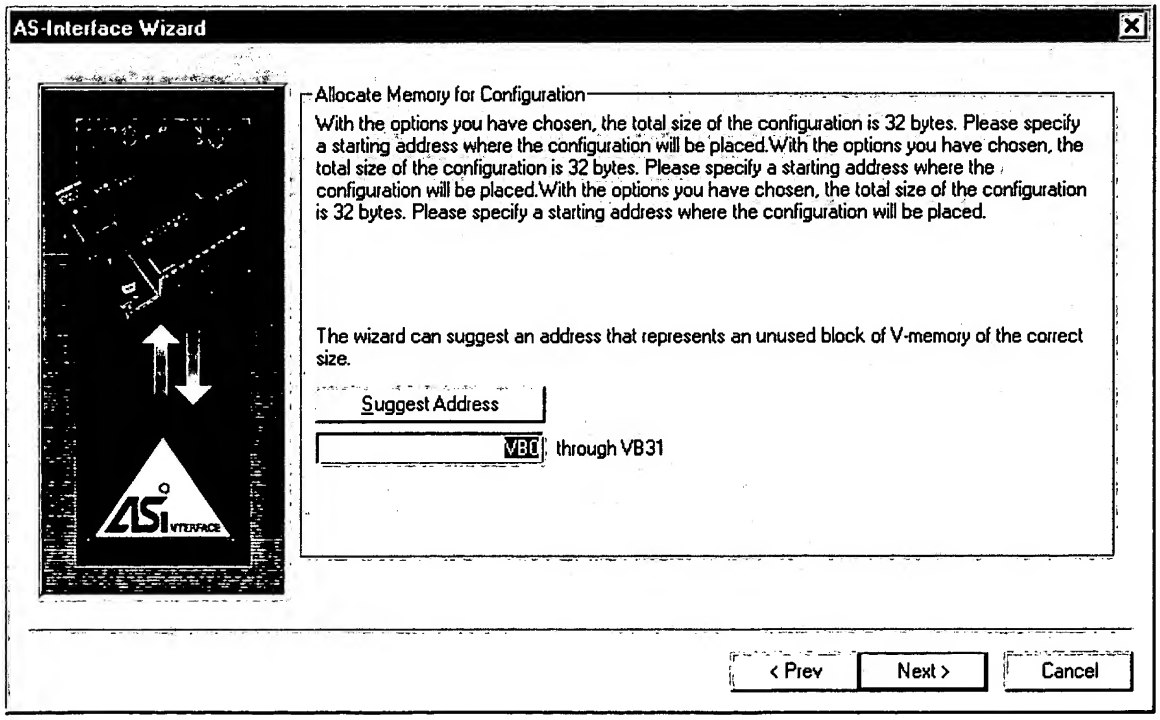
2.8.8 Creation of unique symbol names

Whenever you drop a new slave into the grid, a unique symbol name must be created. Verify that this is so by defining symbols in all the places specified in the duplicate symbol names section.
Verify that

2.8.9 Update

Slave types supported for this grid can be found in the specification.
Verify that all valid slaves for this type can be updated from the controller. This includes the special actuator and sensor types.
Verify correct behavior when an update occurs that includes standard, extended A and extended B slaves.
Verify correct behavior when an update occurs that contains unknown slave types.

2.9 Screen 7



This screen specifies where the configuration data will be place in the controller.

2.9.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,
Verify that the text accurately reports the number of VB locations required for the configuration. This should be run with the minimum sized configuration as well as the maximum and a scattering of samples between.

2.9.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the NEXT button is active

2.9.3 Previous button

Verify that the previous button takes you back to screen 2

2.9.4 Next button

Verify that the next button takes you to the next screen

2.9.5 Cancel

Verify that the cancel button and the Window X button bring up the cancel failsafe dialog.
Verify that the wizard cancels gracefully.

2.9.6 Address box

Verify that only VB addresses can be specified. Try VW, QB, and a few others, both valid addresses and invalid addresses.
Verify that only addresses that are within range for the target controller can be specified. Try this with different controllers.
Verify that the correct ending address is displayed based on the starting address and the number of bytes required.

2.9.7 Suggest address button

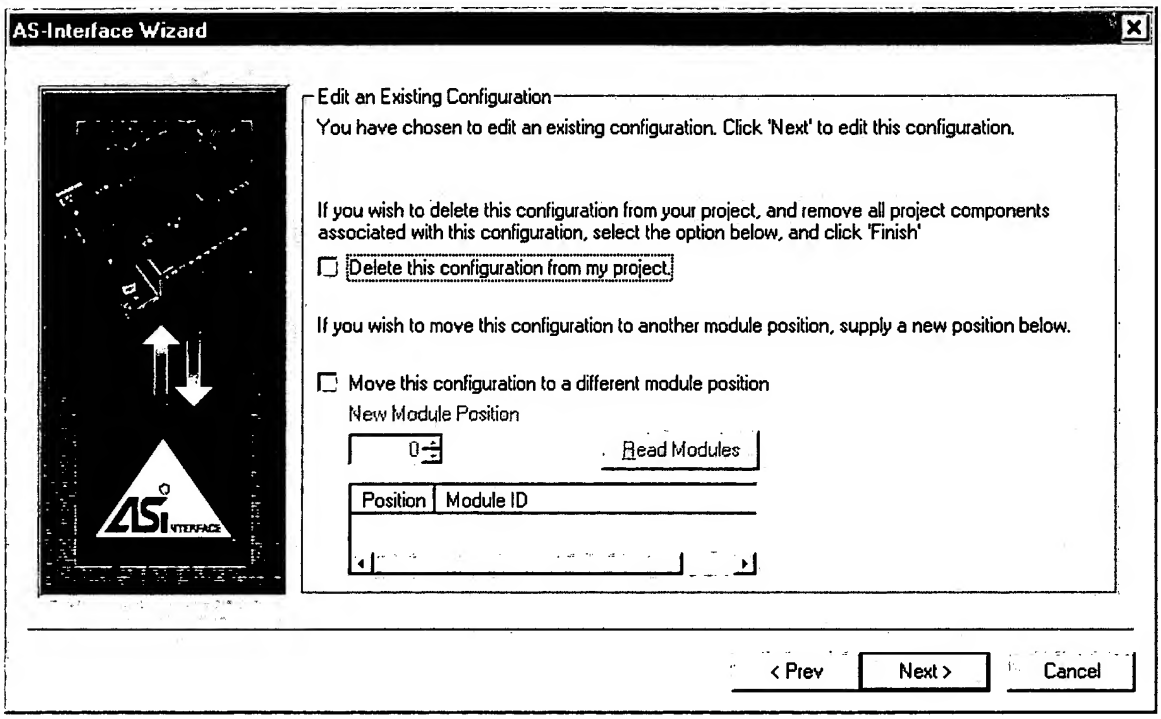
The suggest address button will run through the cross reference and look for holes large enough to fit the configuration
Create a few situations in which a number of holes, too small to fit, must be skipped over in order to find a fit.
Attempt the case in which no holes could be found.
Hit suggest a number of times until end of memory is reached. Verify that this is reported and that wrapping occurs and it begins again from the first VB locations.

3 Reload tests

3.1 Configuration verification

Verify that configurations created by this wizard can be reloaded without error.
Verify projects that have AS-I generated data can be saved and loaded correctly
Configurations must be saved/loaded as well as downloaded and uploaded.

3.2 Screen 8



3.2.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that test is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,
Verify that the text accurately reports the number of VB locations required for the configuration. This should be run with the minimum sized configuration as well as the maximum and a scattering of samples between.

3.2.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the NEXT button is active

3.2.3 Previous button

Verify that the previous button takes you back to screen 2

3.2.4 Next button

Verify that the next button takes you to the next screen

3.2.5 Cancel

Verify that the cancel button and the Window X button bring up the cancel failsafe dialog.
Verify that the wizard cancels gracefully.

3.2.6 Delete configuration

Verify that when you check the delete button that the NEXT button changes to FINISH.
Select delete and press finish. Verify both yes and no cases of the finish dialog

Select delete and then cancel out of the wizard, be sure the configuration is not damaged in any way.
Create at least three configurations, be sure the desired configuration is deleted without damaging the other two configurations.

3.2.7 Move configuration

Verify that when the Move box is unchecked that the position spinner and the read operations are blocked
Verify that when the Move box is selected that the position spinner and the read operation are allowed.
Move a configuration from each of the positions and verify that it is moved correctly.
Symbol names for each configuration will be retained upon move.
POU names will be changed.
Make sure that everything is moved. No pieces should be left behind.
Verify that a configuration cannot be moved into a position that already contains a configuration.

3.2.8 Position spinner

Verify that the position spinner is bounded by 0 at the lower end and the maximum number of expansion I/O-1 per CPU on the upper end. Try with all supported CPU types.

3.2.9 READ module

Verify the case where no modules are present
Verify that the function correctly identifies Asi masters. For example, configure some modules that kind of look like Asi, like an 8DI/8DO or an 8AI/8AO by themselves.
Find one module at each position
Verify multiple ASI masters found. Fill the whole thing with ASI.

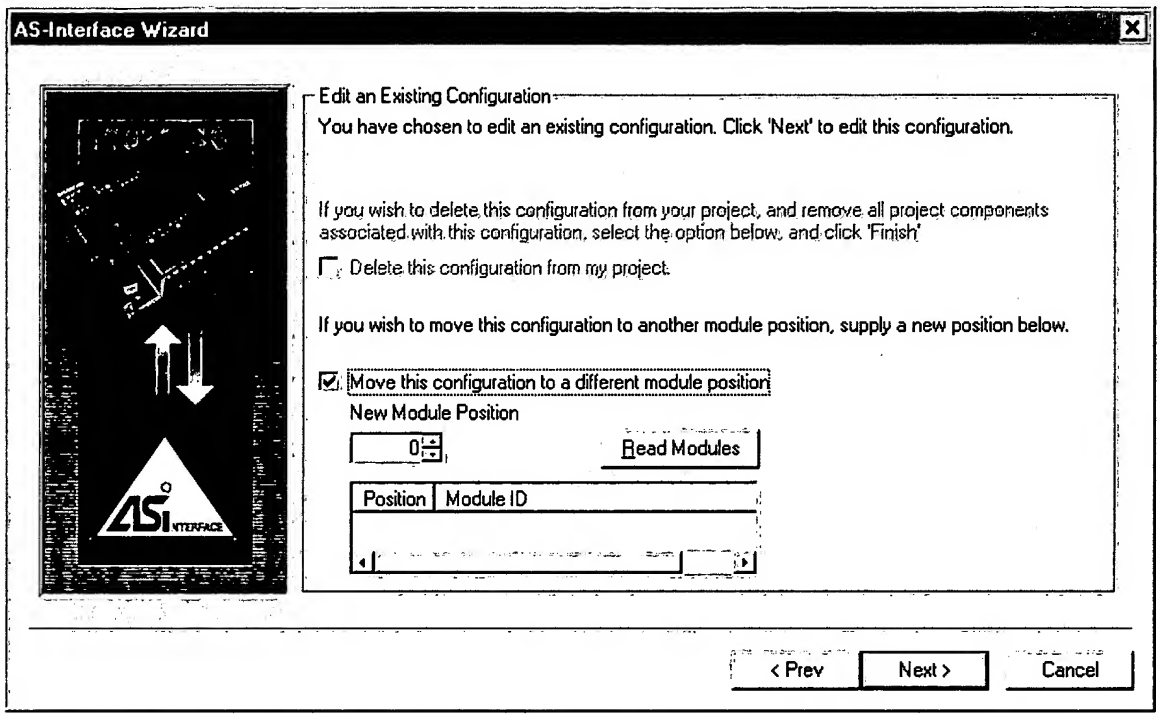
3.2.10 Suggest address button

The suggest address button will run through the cross reference and look for holes large enough to fit the configuration
Create a few situations in which a number of holes, too small to fit, must be skipped over in order to find a fit.
Attempt the case in which no holes could be found.
Hit suggest a number of times until end of memory is reached. Verify that this is reported and that wrapping occurs and it begins again from the first VB locations.

3.3 Deleting configuration

Verify that delete deletes the configuration and all associated with it
Verify that when you cancel with delete selected that nothing is deleted
Verify that other configurations are not damaged when you delete a configuration

3.4 Moving configuration



Move a configuration from each position to each position.
Move a configuration from between two other configurations
Move a configuration into a hole between two other configurations
Attempt to move a configuration into a spot that is already occupied
Make sure that when the configuration is moved, that the location occupied previously is cleaned up correctly.

4 Compare tests

The compare function must be able to detect the following situations:

4.1 Command and response bytes

Verify with command and response byte match, also AIW/AQW
Verify with command byte different
Verify with response byte different
Verify with AIW different
Verify with AQW different
Verify with all different

4.2 Slave match/mismatch

Slave in Micro/WIN matches the one in the master
Slave in Micro/WIN but missing in master
Slave in master, but missing in Micro/WIN
Slave in Micro/WIN is different from that in master

4.3 Communication disruption

Verify case handled where master ready bit is not set
Verify case where command acknowledge never comes high
Verify error responses, both known and unknown are handled correctly

Verify that compare handles communications disruptions OK.(cables pulls et. al.

5 Update tests

Verify that update handles unknown slaves gracefully
Verify that update does not overwrite existing slave definitions. Warnings should be posted telling that a slave was not updated.
Verify update works correctly for all slave types in all slave positions.

6 Code Generation tests

6.1 Symbol tables

Verify that symbol tables are created correctly for all slave types in all slave positions. This includes addresses as well as symbol names

6.2 POU elements

Verify that any POU elements are created correctly. Possible POU elements include:

6.2.1 ASIx_CTRL

This subroutine is responsible for the transfer of I/O data to and from the master.

6.2.2 ASIx_READ

This subroutine will read the specified bank from the master. This is pretty straight forward, the only thing that really needs to be verified is that the correct addresses get placed in to the code.

6.2.3 ASIx_WRITE

This subroutine will read the specified bank from the master. This is pretty straight forward, the only thing that really needs to be verified is that the correct addresses get placed in to the code.

6.3 POU comments

Verify that all POU comments are accurate and that there are no grammatical or spelling problems.

6.4 Data block

Verify that no entries are added to the data block and that this wizard does not touch the data block in any way.

6.5 IEC

Verify that all generated POUs compile correctly when running in IEC mode.

6.6 Analog input filtering

Verify that the correct analog inputs have filtering turned off.

7 Functional tests

7.1 Discrete transfer

Discrete transfers are handled by the _CTRL block. Whenever discrete transfers are required, be sure that the network to handle them is present in CTRL.
Standard and A slave transfers are accomplished by communicating with Bank 0 of the AS-I master.

7.2 Extended address B transfer

Discrete slave type B transfers are handled by the _CTRL block. Whenever discrete transfers are required, be sure that the network to handle them is present in CTRL
Slave B transfers are accomplished by communicating with Bank 31 of the AS-I master.

7.3 Analog transfer

Analog transfers are handled by the _CTRL block. Whenever an analog transfer to a specific slave is configured in the grid, additional code is added to the analog section of the block. Verify that these sections are added correctly.
Analog transfers are accomplished by communicating with Banks 32-48 of the AS-I master.

7.4 AS-I Read

The _READ block allows user read access to any bank within the controller.
Verify that error codes are handled correctly.
Verify that this block is functioning as expected.

7.5 AS-I write

The _WRITE block allows user read access to any bank within the controller.
Verify that error codes are handled correctly.
Verify that this block is functioning as expected.

8 Change slave address tests

The change slave address screen allows customers to reprogram their slave addresses for online slaves.
Verify that this screen cannot be invoked unless communications to the CPU are functioning.
On entry to this function, the following screen should be displayed.

8.1 Appearance

Verify that the picture is correct and is located correctly within the window.
Verify that text is technically and grammatically correct.
Verify that no spelling errors exist.
Verify that no truncation problems exist,

Verify that the text accurately reports the number of VB locations required for the configuration. This should be run with the minimum sized configuration as well as the maximum and a scattering of samples between.

8.2 Navigation

Verify tab and arrow navigation is correct.
Verify that upon entry, the NEXT button is active

8.3 Previous button

Verify that the previous button takes you back to screen 2

8.4 Next button

Verify that the next button takes you to the next screen

8.5 Cancel

Verify that the cancel button and the Window X button bring up the cancel failsafe dialog.
Verify that the wizard cancels gracefully.

8.6 Delete configuration

Verify that when you check the delete button that the NEXT button changes to FINISH.
Select delete and press finish. Verify both yes and no cases of the finish dialog
Select delete and then cancel out of the wizard, be sure the configuration is not damaged in any way.
Create at least three configurations, be sure the desired configuration is deleted without damaging the other two configurations.

8.7 Move configuration

Verify that when the Move box is unchecked that the position spinner and the read operations are blocked
Verify that when the Move box is selected that the position spinner and the read operation are allowed.
Move a configuration from each of the positions and verify that it is moved correctly.
Symbol names for each configuration will be retained upon move.
POU names will be changed.

8.8 Communications tests

Verify that broken communications is detected and handled correctly.
Verify correct behavior when the master ready bit is off
Verify correct operation whenever the command active bit fails to turn on
Verify correct response to error codes, both known and unknown errors.

8.9 Slave detection

Verify that this window correctly detects all slave types in all positions.
Verify for a variety of master positions
Verify the case when all slaves are programmed
Verify the case when no slaves are programmed
Verify only Standard
Verify only A
Verify only B
Verify only analog

8.10 Slave moving

Verify that standard can only be moved into standard locations(A/B check box off)
Verify that A slaves can be moved into either A or B space

Verify that B slaves can be moved into either A or B space
Verify that analog slaves and only be moved within standard slave space
Verify that a slave cannot be moved on top of itself
Verify that a slave cannot be moved on top of an existing slave

9 Ozzy

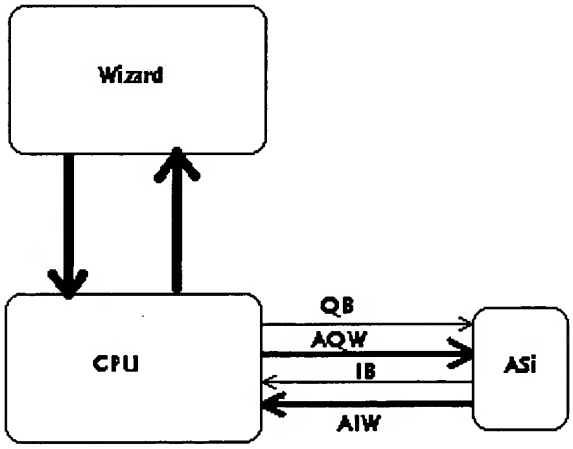
9.1 Introduction

Functionality of the AS-i wizard includes the ability to read and identify slave modules from the master. For an exhaustive test of this functionality, a minimum of X different slaves need to be purchased in order to verify that the Wizard correctly identifies all slaves. In addition, many different slaves that are not supported need to be purchased.
The wizard can read up to 31 non extended and 31 additional extended slaves. This would require the purchase of 62 slaves, power supplies, and wiring to achieve comprehensive coverage.
The cost to purchase and setup of all these slave is prohibitive, therefore an AS-i master simulator was created.

The AS-i master simulator (Ozzy) connects to a CPU226XM with special firmware and allows the tester to simulate every possible slave located at every possible slave address. Up to three Masters can be simulated at the same time.

9.2 Hardware requirements:

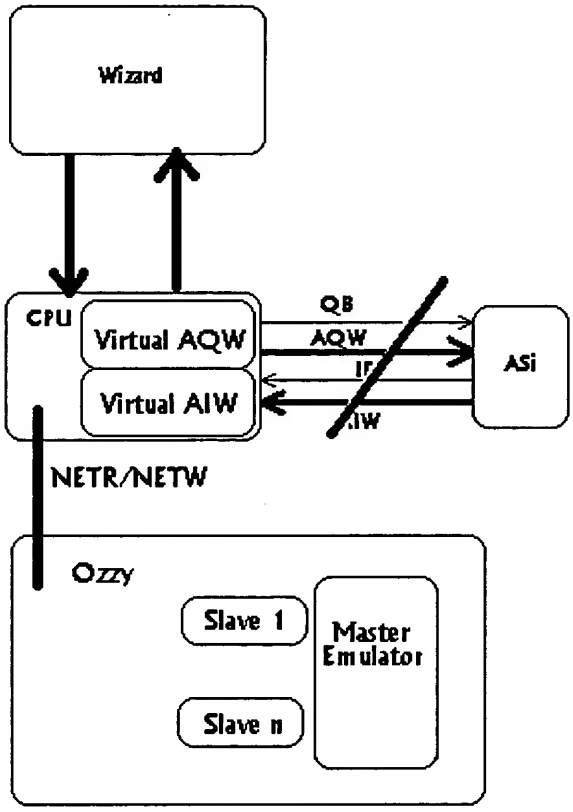
Traditional AS-i configuration



Traditional ASi configuration

In a traditional AS-i configuration, an AS-i master is connected to the S7-200 CPU as a pair of I/O modules. Micro/WIN sends commands and data to the AS-i master using eight words of AQW locations and a byte of QB memory.
The AQW locations contain the commands, typically written to bank 2, and the QB byte is used as a command byte to tell the master to process a request.
Micro/WIN then reads the command handshaking from the IB byte and the data from the AIW words.
Some commands process with a couple of iterations, some require many iterations.
The program in the S7-200 also communicates with the AS-i master using this interface.

AS-i(Ozzy) simulator configuration.



The AS-i master simulator works by intercepting the command requests from the wizard, processing them, and returning a response.

The AS-i master simulator (Ozzy) exists as an application program in a CPU226XM controller. This program cycles through a process of reading QB and AQW from the test controller, processing any commands, and writing back to IB and AIW in the test controller. This cycle of events occurs as quickly as possible. A typical cycle takes approximately 50 milliseconds.
Communications with the CPU under test is performed using NETR and NETW instruction from port 1 of Ozzy to port 1 of the test CPU.

Since AQW and AIW memory are accessed directly on the I/O bus by the S7-200, there is no way to modify these locations easily. A special release of the CPU226XM firmware(09.23) was provided that maps all accesses to AQW and AIW into an image that resides from VB8000-8127

Whenever a communications telegram or the program within the CPU writes to AQW0-63 memory, the information is also saved in VB8000-8063.
Whenever a communications telegram or the program within the CPU reads from AIW0-62, the information stored at VB8064-8127 is returned instead.

A master pointer table is kept starting at VB8130. This table contains the QB, IB, AQW, and AIW addresses associated with each master. This table is read every cycle and address offsets for each master are calculated. Breakout of this memory range are as follows:
VB8130 - Master 0 QB number
VB8131 - Master 0 IB number
VB8132 - Master 0 AQW number
VB8133 - Master 0 AIW number

VB8134 - Master 1 QB number
VB8135 - Master 1 IB number
VB8136 - Master 1 AQW number
VB8137 - Master 1 AIW number

VB8138 - Master 2 QB number
VB8139 - Master 2 IB number
VB8140 - Master 2 AQW number
VB8141 - Master 2 AIW number

9.3 Sequence of events:

User asks for Read modules operation
- MicroWin sets AQW0 to 0x1400
- MicroWin sets QB0 = 0xC2
  - CPU copies AQW0 to VW8000
    - Ozzy Reads VB8000-8063 from the CPU
    - Ozzy reads QB0-15 from the CPU
    - Ozzy determines that a version ID request is asked for
    - Ozzy copies the version ID information into its bank 2
    - Ozzy copies the version ID information into AIW0-15
    - Ozzy writes VB8064 to 8127 to the CPU
    - Ozzy writes operation complete to IB0
- MicroWin reads IB0 and sees that the operation is complete
- MicroWin reads AIW0-15, CPU supplies values in VB8064-8079
- MicroWin displays that a master is located

9.4 Instructions that are supported by Ozzy include:

Master number is implied by the fact that the QB is set for each master.

Read Version ID 0x14
This operation is sent by Micro/WIN during a read modules request. It is sent to bank 2. Ozzy will respond by transferring its pre-loaded version string into both Bank 2 and the AIW response buffer.

9.4.1 Write version ID 0x15

This is a test operation. Ozzy will detect this command and transfer the version string contained in the AQW buffer into its internal storage area for the string.

This command will be used to load version strings into Ozzy for verification of read modules
The AQW buffer is formatted as follows:
AQW0 - 0x15 - Command
AQW2 -'CP'
AQW4 - ' 2'
AQW6 - '43-
AQW8 -'-2'
AQW10 -' V'
AQW12 -'1.'
AQW14 'xx'

Write Command status 0x16
Whenever Micro/Win sends a command, a command status is returned in the second byte of the AIW area. This byte can contain a 0(no error) or an error code. This command is used to set the command status that will be returned whenever a request for a command is issued. The automated test tools will use this to verify that all Micro/WIN generated Asi commands react correctly to error responses.
The AQW buffer is formatted as follows:
AQW0 - 0x1600
AQB2 - Resp (byte value(0-255))

9.4.2 Process Get Extended Total Configuration 0x39

Whenever Micro/Win needs to know which slave are present in the system, it sends this command. Information about every possible slave in the system is returned. See the spec for more details. A lot of information is returned, so much in fact that it takes multiple accesses across multiple banks to return it. Upon receipt of this command, Ozzy will copy the saved slave configuration into the bank memory for the slave. It will also copy the data for bank 2 into the AIW area so that it is immediately available to Micro/WIN.
After MicroWIN has get a no error command status, it will toggle QB0 from 0xC2 to 0x82 and read in the information about Bank 2. It will then increment QB0 to 0x83 and read in the information from Bank 3. This continues until Micro/WIN determines that it has read in all information about the slaves in the system.
Whenever will read this QB location and copy the information for the specified bank into the AIW area.

The AQW buffer for the first access is formatted as follows:
AQW0 -0x3900

9.4.3 Initialize master 0xFF

Before a master can be accessed by Micro/WIN, it must first be created in Ozzy. This is accomplished when the automated test sends an 0xFF to Ozzy.
Upon receipt of this command, Ozzy will create the specified master in memory and initialize all of its data structures.

Format for this command is:
AQW0 - 0xff00

9.4.4 Bank programming.

Since a bank consists of 16 bytes and two bytes have been used for command processing, the standard command mechanism will not function very well for programming large quantities of data into the Ozzy

master banks. Therefore, by decree, whenever a bank address greater to or equal than 32 will implicitly specify a write bank operation.
The bank to be written to is: Specified bank - 32
Example, if automated test program specifies bank 32, bank 0 will be programmed. If bank 33 is selected, bank 1 will be written to.

Sequence of events for initialization:
Module is created in the controlled using indirect addressing to access SD memory
Master pointer are specified to Ozzy via VB8130-8146
Ozzy master is created with command 0xff
Ozzy version number is programmed using command 0x15
Ozzy command status is set using command 0x16
Ozzy slaves are created using bank access
Ozzy master is now ready to use.